RICE UNIVERSITY

Space–Time Finite Element Computation of the Aerodynamics of Flapping Wings

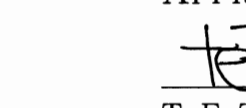
by

Bradley J. Henicke, 2nd Lt, USAF

A Thesis Submitted
in Partial Fulfillment of the
Requirements for the Degree

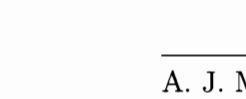
Master of Science

Approved, Thesis Committee:

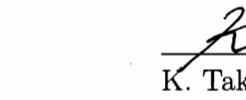


T. E. Tezduyar, Chair
Professor of Mechanical Engineering and
Materials Science

J. E. Akin
Professor of Mechanical Engineering and
Materials Science and Professor of
Computational and Applied Mathematics



A. J. Meade
Professor of Mechanical Engineering and
Materials Science



K. Takizawa
Associate Professor in Department of
Modern Mechanical Engineering and
Waseda Institute for Advanced Study
Waseda University, Tokyo, Japan

Houston, Texas
April 2011

The views expressed in this thesis are those of the author and do not reflect the official policy or position of the United States Air Force, Department of Defense, or the U. S. Government.

Abstract

Space–Time Finite Element Computation of the Aerodynamics of Flapping Wings

by

Bradley J. Henicke

The details of the aerodynamics of flapping flight continue to pose a considerable challenge to a complete understanding of flight. Advanced computational fluid mechanics technology coupled with experimental data offers a unique perspective into these aerodynamics. The difficulty of computing such complex aerodynamics is mostly related to the presence of moving and deforming solid surfaces. The finite element method with the Deforming-Spatial-Domain/Stabilized Space–Time (DSD/SST) formulation, which was developed by the Team for Advanced Flow Simulation and Modeling for the computation of flow problems involving moving boundaries and interfaces, is well-suited for this type of problem. The DSD/SST method is further enhanced with a variational multiscale turbulence model and other special techniques, which were developed in the context of the DSD/SST method for flapping flight computations and involve temporal NURBS basis functions. These techniques are applied to the computation of locust flapping flight, where the prescribed motion and deformation of the wings are based on digital data extracted from wind tunnel experiments. This forms a foundation upon which further study may reveal additional insight into flapping flight aerodynamics.

Acknowledgments

It has been said that no one reaches great levels of success entirely on their own. Therefore, it is only right that I acknowledge those who have played a role in the success of this research.

I would like to begin by thanking Dr. Kenji Takizawa, who made this all possible through his diligence and continuous hard work. When I thought it could not be done, you encouraged me to keep trying—even all the way from Japan. I wish you all the best as you return to Japan to teach at Waseda University in Tokyo.

Next, I would like to thank Dr. Tayfun Tezduyar who provided me with the opportunity to pursue graduate studies at Rice University and work with T★AFSM. I appreciate your guidance and tireless efforts to ensure my work is the best it can possibly be. The opportunities you have provided and lessons I have learned through this experience will not soon be forgotten.

I would also like to thank Dr. Ed Akin and Dr. Andrew Meade for serving on my thesis committee.

During my time at Rice, I have had the privilege to conduct research with some of the finest team members for which anyone could ask. Much of this research was made possible by the efforts of Tony Puntel and Darren Montes. Both of you went above and beyond what was asked. The research presented in this thesis is also a reflection of your hard work and dedication. To Tim Spielman, Tyler Brummer and Tracee Curlett—thank you for all you have done to help me over the past two years. I wish you all the best as we close this chapter and move on to the next. I would also like

to thank Creighton Moorman, Eddie Wright, Kat Schjodt, Nik Kostov, Matt Fritze and Matt Guertin with whom I have had the privilege to work at some point during the last two years.

From the beginning, my family has encouraged me to use my God–given talents to do the best I can in everything that I do. I would like to take this opportunity to thank my mom, dad, sister, brother and extended family for supporting me and being there when things were going well and when they were not.

This work was supported by NSF Grant CRCNS-0903949. Method development and evaluation components of this work were supported also in part by ARO Grant W911NF-09-1-0346. Computational resources were provided in part by the Rice Computational Research Cluster funded by NSF Grant CNS-0821727. I would like to thank Professor Fabrizio Gabbiani and Dr. Raymond Chan (Baylor College of Medicine) for providing us with the digital data and photos extracted from the videos of the locust in their wind tunnel.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Inspiration from nature offers tremendous potential for innovation in many fields, including flight. In nature, flight is often achieved by means of flapping wings, which has fascinated man for much of history; one may be reminded of the mythological story of Icarus who achieved winged flight—though his flying exploits were admittedly short-lived. This curiosity, however, is more than simply a fleeting thought; rather, as noted by Wu, such curiousity is essential to the "formation of sound physical conceptions, successful developments of required mathematical tools, and novel engineering" [84].

While such curiousity is important, today's technology has made the need for a broader knowledge of nature's means of flight more immediate. Such knowledge is now directly applicable to the development of micro air vehicles (MAVs), which are defined as air vehicles that have no length dimension greater than 15.24 cm and an approximate gross takeoff weight of 200 g or less [36]. According to Mueller, MAVs have the capability of accomplishing special, limited-duration military and civil missions that would be impractical for larger aircraft; MAVs may even assist with hostage rescue and counter-drug operations [34]. The design of these vehicles poses a new set of obstacles, which may potentially be overcome by learning from nature.

Though much knowledge has been gained about flapping flight, the means by which in–flight collision is avoided in nature remains an area of ongoing research. To avoid collisions, a flying insect or other animal must not only sense the pending stimulus, but it must also respond appropriately in a very short period of time. In addition to MAV applications, enhancing our knowledge of this behavior may lead to advancements in other aerospace and biomedical applications, such as artificial vision.

The collision avoidance response is quite complex in that it must bridge the neural and muscular systems to produce mechanical and thus, aerodynamic effects. Studying collision avoidance, therefore, necessitates a highly-integrated approach if we hope to gain a further understanding of in-flight collision avoidance behavior. Working in collaboration with the Baylor College of Medicine (BCM) and the University of Arizona, the Team for Advanced Flow Simulation and Modeling (T★AFSM) is using computational fluid mechanics to better understand the mechanical and aerodynamic aspects of collision avoidance behavior. Specifically, this computational analysis is based on the Deforming-Spatial-Domain/Stabilized Space–Time (DSD/SST), which was introduced by T★AFSM for flow problems with moving boundaries and interfaces and has been applied to a 3D computation of flow past a pair of flapping wings [33]. For improved accuracy, T★AFSM has recently developed a number of improvements to this formulation. These improvements include the addition of an advanced turbulence model, which is a space–time version of the residual–based variational multiscale method (VMS), and an increase in polynomial power for the basis functions of the spatial discretization and time integration through the use NURBS.

Prior to delving further into the details of the computational method, it is worthwhile to review flapping flight aerodynamics and the broader context of this research effort. These reviews will be covered in Sections 1.1 and 1.2. Additionally, for the reader who may be unfamiliar with NURBS, an overview of NURBS and associated

terminology is provided in Section 1.3.

1.1 A Brief Introduction to Flapping Flight

Even the reader who is being introduced to flapping flight for the first time may be familiar with the well-known story that has circulated since at least the 1930s of the aerodynamicist who "proved" over a dinner conversation that a bumblebee is incapable of flight. The aerodynamicist's quick calculations were based on the assumption that the bee's wings would behave like smooth, flat plates at low Reynolds numbers over which flow is likely to separate early due to its laminar nature, generating insufficient aerodynamic forces for flight. As when first proposed, life experience should quickly cause one to question the outcome of this "proof," and there now exists a much better understanding of flapping flight that disproves the validity of these assumptions [32]. Nevertheless, this story illustrates an important point regarding flapping flight: flapping flight is more complex than it may initially seem. Despite this complexity, a basic understanding of the subject may still be acquired by reviewing the results of previous research.

It should be noted that this section is not meant to serve as an extensive review of the flapping flight aerodynamics. Entire books have been written on this topic and an attempt to provide a comprehensive review would not be appropriate here. Thus, this section is meant to provide only a basic working knowledge of flapping flight to better understand the content of this thesis. The remainder of this section is largely a summary of Chapter 4 in [4]. For a more information, the reader is encouraged to refer to this book or another one of numerous other works available on this topic.

For simplicity, let us begin by restricting this discussion to the case of steady, level, unaccelerated flight of an animal capable of flapping flight. In this case, the flapping wing must generate both a lift force to oppose the force of gravity and thrust to oppose the drag force caused by the animal's motion through the air. This differs from most

fixed–wing aircraft, where in general terms, the lift force is generated by the wings and the thrust is generated by one or more engines, and such an integrated approach to flight suggests that the wing motion is much more complicated than a simple up–and–down flapping. A trace of wingtip motion relative to the body will reveal that this motion varies significantly among different animals. Upon further examination, it can also be observed that the wing is not, in fact, a flat plate; instead, the wing generally deforms throughout the motion of a single flap cycle. Moreover, the relative wind felt at a point on the leading edge of the wing is also a combination of the wind due to the animal's forward velocity and the velocity due to the wing's flapping motion.

The motion of a single flap cycle can be further divided into a downstroke and an upstroke. The downstroke, when the wing is moving down and forward, generally produces most of the lift and thrust. During this part of the cycle, the section of the wing near the body tends to produce more lift while the tip produces more thrust due to the varying angle of attack and relative wind across the span. The degree to which lift and thrust are generated on the upstroke (when the wing moves up) varies by animal. In general, however, the downstroke motion and deformation are different from that of the upstroke.

It is also important to consider the flight regime of these animals and the effects of unsteady aerodynamics. Flapping flight generally occurs at relatively low speeds with fairly small length scales; thus, the Reynolds number of a bird is much lower than that of modern transport airplanes. In addition to the kinematic differences of flapping flight, this lower Reynolds number results in potentially different flow behavior compared to most aircraft. Previous research has also suggested the importance of unsteady aerodynamic effects, such as the clap-fling mechanism and dynamic stall. The clap-fling mechanism, which is attributed to Weis-Fogh in [4], helps to overcome the Wagner effect, or the initial delay in lift production. The wings are clapped to-

gether at the end of the upstroke and then are peeled apart beginning at the leading edges to start the downstroke. Air rushing between the wings creates a bound vortex and results in almost immediate lift production. Rotational dynamic stall generally occurs when the wings quickly rotate to higher angles of attack. Similarly, in the case of translational dynamic stall, the wing begins the flapping motion at an angle much greater than its stall angle. These effects result in much greater, albeit transient, aerodynamic force generation, and it is expected that they play an important role in collision avoidance maneuvers.

1.2 Project Overview

While many species exhibit flapping flight in nature, the locust has been selected as the subject of this collision avoidance behavior study. The locust, which can be seen in Figure 1.1, is readily-available, possesses strong flying skills as noted in [80] and has previously served as the subject of well-documented experiments. For this study, the locusts' ability to fly in large swarms is also important, which can be seen in Figure 1.1.

Figure 1.1: Sketch of a locust and photograph of a locust swarm from [19, 31].

The locust possesses two sets of wings, designated the hindwings and forewings, that are each capable of wide range of motion and simultaneous deformation during flight. Wing deformation, particularly in the hindwing, is also not trivial in that it varies along the chordwise and spanwise-directions of the wing and is time-dependent.

Additionally, a phase-lag exists between the motion of the hindwing and forewing. The locust's forewing and hindwing can be seen in Figure 1.2.

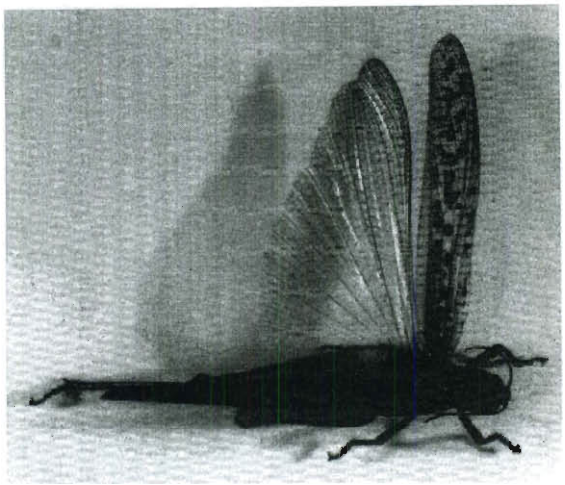


Figure 1.2: Photograph of locust forewing and hindwing from [37].

In 1956, Weis-Fogh and Martin Jensen published an extensive study of locust flight [82, 80, 27, 81]. In their study, they concluded "neither the kinematics nor the dynamics are sufficiently well known to permit a theoretical treatment of the energetics of natural flapping flight" [82]. Since that time, a number of studies have been conducted through which we have gained a better understanding of locust flight. More recently, Wootton et al. and Herbert et al. conducted experimental and computational studies to better understand the structure and function of the locust hindwing [83, 20]. Walker et al. studied locust wing kinematics and deformation to better understand the importance of these characteristics [79]. In 2009, Young et al. used experimental data in computations to study locust straight-flight behavior. In this study, it was shown that "the details of insect wing topography and deformation are important aerodynamically" [85]. Photos of wind tunnel smoke visualizations from this study, which clearly show vortex structures in the wake of the locust, can be seen in Figure 1.3. The nature of the collision avoidance response requires an integrated approach for which a collaborative research effort is appropriate. Experimentation is being conducted at the University of Arizona and the Baylor College of Medicine.

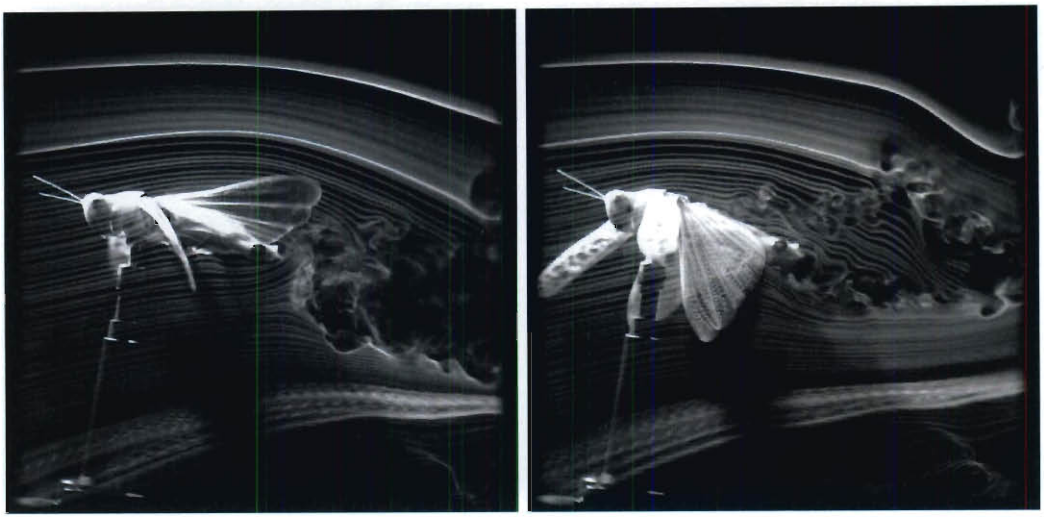


Figure 1.3: Wind tunnel smoke visualizations of locust flight from [1, 35].

Efforts at the University of Arizona, led by Dr. Sergey Shkarayev, focus on the aerodynamics of straight flight; meanwhile, efforts at the Baylor College of Medicine, led by Dr. Fabrizio Gabbiani, focus on the linked neurological and aerodynamic aspects of collision avoidance behavior. T★AFSM at Rice University is using the data collected by the other collaborators to conduct computational fluid mechanics simulations to better understand these behaviors. This National Science Foundation research project, entitled "Collaborative Research: Integrated Analysis of In-Flight Collision Avoidance Systems," began in September 2009.

The potential for asymmetric motion and deformation in the collision avoidance response contributes additional complexity to these simulations. Thus, prior to studying collision avoidance behavior, T★AFSM had to first construct and validate a flapping flight computational model. Based on previous research, it is clear that a reliable means of replicating wing motion and deformation is crucial for achieving an accurate fluid dynamics simulation from which conclusions regarding locust flight may be drawn. While simple mathematical models may be used to describe the motion and deformation of the wings, they are quite difficult to generate at the desired level of accuracy. Therefore, even for initial testing, it is preferable to use experimental

data to describe wing kinematics. Additionally, using experimental data as an input to computational simulations represents an effort to integrate the experimental and computational fields of research in aeronautics.

For generating the computational model and testing of techniques, T★AFSM created test cases based on experimental data provided by collaborators at the Baylor College of Medicine. Although the gathering of experimental data is largely under the purview of our collaborators, it is useful to provide an overview of the source of this data as it contributes directly to the degree to which the simulations accurately represent locust flight. Experimental data is acquired by placing a live locust, which has been collected from the wild, in a low-speed wind tunnel using a tether system. To simulate forward flight, the velocity of the wind tunnel is then increased to 2–5 m/s during which time the locust begins flapping. A tether system is used, which includes a counterweight and allows for 6-degree–of–freedom motion. However, to collect the data to be used in simulations, a rigid tube is added to the tether system for increased stability keeping the locust in the camera field of view. Two high-speed video cameras placed above the tunnel record the motion of the wings at 500 frames/s. Based on the data collected from these cameras, the movement of tracking points on the locust in 3D space can be attained using a process known as photogrammetry. Additional smoothing may be applied to the data prior to its being provided to T★AFSM for simulations. This data forms the basis of the computational efforts presented in this thesis.

1.3 A Brief Introduction to NURBS

As discussed in Section 1.2, previous research has shown the importance of wing kinematics and deformation in accurately simulating locust flight. Thus, an accurate representation is also essential to obtaining an accurate fluid dynamics simulation based on a prescribed motion of the wing structure. Non–uniform rational B-splines,

or NURBS, are one tool that can be used to obtain an accurate prescribed motion in flow computations. Although NURBS have existed for some time, their use in analysis, such as in computational mechanics, is relatively new and is currently being advanced in the developing field of isogeometric analysis.

Similar to the preface given in Section 1.1, this section is not meant to serve as an exhaustive explanation of NURBS. The interested reader is encouraged to seek additional information from the many books and papers, such as [23, 9, 8, 11, 18], published on the topic. The information presented in this section is largely a summary of Section 1.4 and Chapter 2 of [18]. Despite the limited nature of this summary, it should still prove useful to the reader as an introduction or review of NURBS and associated terminology.

NURBS have historically been used to create representations of geometric objects and perhaps it is in this context that they may be most easily introduced. Let us begin by assuming a geometric object, such as a curved surface, exists for which there is a need to create a model. This model may be created by defining the location of points on the surface of that object, connecting those points, and interpolating to create a continuous surface. If the original object existed in 2D, such as in the case of the curved surface, and linear interpolation functions are employed, then the model is a faceted surface where the accuracy with which the model represents the original object is partly determined by the number of points used. While this representation is often times only an approximation, the geometric object may be represented exactly using far less points using NURBS. NURBS offers a number of advantages in terms of continuity, refinement, and increased accuracy as demonstrated in [18]. However, the means by which this model is generated using NURBS may seem less straightforward.

A basic understanding of NURBS must also include an introduction to NURBS terminology. In NURBS, an object is defined by control points. However, these control points in many cases do not lie on the surface of the object meaning they are not

interpolatory. The control mesh connects and interpolates the control points. Each control point also has an associated weight that may be thought of as representing the effect on the surface of that particular control point. If all control points are equally weighted, then the representation is referred to as a B-spline, or Bézier-spline, rather than a NURBS representation. The representation defined by the control points can be divided by knots which appear as points, lines or surfaces in 1D, 2D or 3D, respectively, and the area between the knots is known as the knot span. The set of knots in one parametric direction is called the knot vector. Multiple sets of control points, or patches, appear as rectangles in 2D and cuboids in 3D and may be combined to represent more complex shapes. These terms are illustrated for a multiple patch, quadratic B-spline surface in Figure 1.4.

Figure 1.4: Two quadratic B-spline surface patches (blue and green) in physical space with knots (orange) as defined by control points (red), which also define the control mesh (black). Note: Additional space added between patches for emphasis.

In the interest of simplicity, these terms have been described in physical space, but the reader should also be aware that there exists a corresponding parameter space and index space. A few additional concepts should be introduced for which the existence of parameter space must be established. The interpolation of the control points occurs in parameter space with basis functions of polynomial order, p. Knots, though they

may be represented in physical space, are defined in parameter space where they may be repeated thereby increasing that knot's multiplicity, m. An entry appearing twice, for example, in the knot vector is said to have a multiplicity of 2. Continuity of the basis functions, one of the previously mentioned advantages of NURBS, is a function of both p and m. Within the knot spans, the basis functions are C^{∞}–continuous, but across knots, the basis functions are only C^{p-m}–continuous.

To illustrate, this means that second–order basis functions, which are also referred to as quadratic basis functions, are C^{1}–continuous (C^{2-1}) across knots, or continuous on the function and its first derivative. If a knot is repeated, thereby appearing twice in the knot vector, then the function is only C^{0}–continuous (C^{2-2}) and continuity is only retained on the function—all derivatives are discontinuous. However, if repeated three times, the second–order basis functions are fully discontinuous, or (C^{-1})–continuous, and that makes a patch boundary.

It is possible to insert additional knots and change the location and number of control points in such a way that it does not change the representation of the surface. This is in contrast to a faceted surface where including additional points may make the surface approximation closer to the original geometric object but will also change the representation of that surface. Although limited in scope, the reader should now have a sufficient NURBS background as it relates to the content of this thesis.

1.4 Content Overview

The following chapters will present the work completed on this project to date.

Chapter 2 presents the problem in terms of the applicable governing equations and the finite element formulation, The DSD/SST formulation is presented with a VMS turbulence model, which is used in these computations.

Chapter 3 discusses the various special modeling techniques developed for this problem, namely a NURBS-based temporal interpolation and the Simple-Shape De-

formation Model (SSDM).

Chapter 4 provides the relevant details of a preliminary computation, including setup and results.

Chapter 5 presents a computation with improved temporal representations.

Chapter 6 describes a test computation demonstrating a flapping wing computed using NURBS for both temporal and spatial representations.

Chapter 7 proposes conclusions which may be drawn.

Chapter 2

Governing Equations and Finite Element Formulations

The study of flapping flight aerodynamics is fundamentally a fluid and structural mechanics problem. Specifically, it is a fluid–structure interaction (FSI) problem, where the air, a fluid, flows over the locust's wings and body, a structure. In this study, the motion of the structure is prescribed from experimental data, which allows the fluid mechanics and structural mechanics aspects of the problem to be decoupled. The problem is, therefore, reduced to a fluid mechanics problem governed by the Navier–Stokes equations. It should be noted, however, that the process of accurate structural motion reconstruction is far from trivial, and it will be covered in detail in Chapter 3.

This chapter begins by presenting the governing equations of fluid mechanics applicable to this problem in Section 2.1. Next, the DSD/SST formulation with the VMS turbulence model is described in Section 2.2. Section 2.3 closes this chapter with a discussion of the new generation space–time formulations used in these computations. This information is presented almost entirely as it appears in [49] where it was first introduced.

2.1 Fluid Mechanics Equations

The governing equations of fluid mechanics applicable to this problem are the Navier–Stokes equations of incompressible flow. Let $\Omega_t \subset I\!R^{n_{sd}}$ be the spatial domain with boundary Γ_t at time $t \in (0,T)$. The subscript t indicates the time-dependence of the domain. The Navier–Stokes equations of incompressible flows are written on Ω_t and $\forall t \in (0,T)$ as

$$\rho\left(\frac{\partial \boldsymbol{u}}{\partial t} + \boldsymbol{\nabla}\cdot(\boldsymbol{u}\boldsymbol{u}) - \mathbf{f}\right) - \boldsymbol{\nabla}\cdot\boldsymbol{\sigma} = \mathbf{0}, \tag{2.1}$$

$$\boldsymbol{\nabla}\cdot\boldsymbol{u} = 0, \tag{2.2}$$

where ρ, $\boldsymbol{u}$ and $\mathbf{f}$ are the density, velocity and the external force, respectively. The stress tensor $\boldsymbol{\sigma}$ is defined as $\boldsymbol{\sigma}(p,\boldsymbol{u}) = -p\mathbf{I} + 2\mu\boldsymbol{\varepsilon}(\boldsymbol{u})$, with $\boldsymbol{\varepsilon}(\boldsymbol{u}) = \left((\boldsymbol{\nabla}\boldsymbol{u}) + (\boldsymbol{\nabla}\boldsymbol{u})^T\right)/2$. Here p is the pressure, $\mathbf{I}$ is the identity tensor, $\mu = \rho\nu$ is the viscosity, ν is the kinematic viscosity, and $\boldsymbol{\varepsilon}(\boldsymbol{u})$ is the strain-rate tensor. The essential and natural boundary conditions for Eq. (2.1) are represented as $\boldsymbol{u} = \mathbf{g}$ on $(\Gamma_t)_{\mathrm{g}}$ and $\mathbf{n}\cdot\boldsymbol{\sigma} = \mathbf{h}$ on $(\Gamma_t)_{\mathrm{h}}$, where $(\Gamma_t)_{\mathrm{g}}$ and $(\Gamma_t)_{\mathrm{h}}$ are complementary subsets of the boundary Γ_t, $\mathbf{n}$ is the unit normal vector, and $\mathbf{g}$ and $\mathbf{h}$ are given functions. A divergence-free velocity field $\boldsymbol{u}_0(\mathbf{x})$ is specified as the initial condition.

2.2 DSD/SST Formulation of Fluid Mechanics

The DSD/SST method introduced in [53, 57, 58, 54] may be applied to a variety of problems. These problems include computations of flows governed by the Navier–Stokes equations, which are given for incompressible flows in Section 2.1, with moving boundaries and interfaces. The DSD/SST method was applied to a variety of problems including free-surface and two-fluid flows, fluid–structure and fluid–particle interactions, and flows with mechanical components in fast, linear or rotational relative

motion (see, for example, [53, 57, 58, 51, 12, 33, 5, 28, 13, 29, 52, 41, 54, 67, 72, 73, 56, 63, 62, 74, 65, 64, 66, 39, 75, 68, 76, 42, 45, 70, 77, 78, 71, 47, 46, 50, 69, 49, 44, 48]).

As stated in [43], the space–time computations in the DSD/SST method are carried out for one space–time "slab" at a time, where the "slab" is the slice of the space–time domain between the time levels n and $n+1$. While the basis functions are continuous within a space–time slab, they are discontinuous from one space–time slab to another. The formulation uses the Streamline-Upwind/Petrov-Galerkin (SUPG) [14] and Pressure-Stabilizing/Petrov-Galerkin (PSPG) [53, 59] stabilizations. New generation DSD/SST formulations, with emphasis on increasing the robustness and lowering the computational cost, were introduced by the T★AFSM in [62].

This section begins by describing the space–time variational formulation with the VMS turbulence model used in these computations in Section 2.2.1. After discussing scale separation in 2.2.2, the DSD/SST formulation is extended to arrive at the VMS DSD/SST formulation in Section 2.2.3. A comparison of the final VMS formulation to the original formulation is also provided in Section 2.2.3. The sections that follow are nearly identical to those presented in [49].

2.2.1 Space–time variational formulation

A space–time variational formulation of incompressible flows (see for example [53, 57, 58, 54]) is written over a sequence of N space–time slabs Q_n, where Q_n is the slice of the space–time domain between the time levels t_n and t_{n+1}, and P_n is the lateral boundary of Q_n. The trial and test function spaces are denoted for the velocity and pressure as $\boldsymbol{u} \in \mathcal{S}_{\boldsymbol{u}}$, $p \in \mathcal{S}_p$, $\mathbf{w} \in \mathcal{V}_{\boldsymbol{u}}$ and $q \in \mathcal{V}_p$. Deriving the variational formulation, starts with multiplying Eqs. (2.1) and (2.2) with the corresponding test functions,

integrating them over Q_n, and setting it equal to zero:

$$\int_{Q_n} \mathbf{w} \cdot \rho \left(\frac{\partial \boldsymbol{u}}{\partial t} + \boldsymbol{\nabla} \cdot (\boldsymbol{u}\boldsymbol{u}) - \mathbf{f} \right) dQ - \int_{Q_n} \mathbf{w} \cdot \boldsymbol{\nabla} \cdot \boldsymbol{\sigma} dQ$$
$$+ \int_{Q_n} q \boldsymbol{\nabla} \cdot \boldsymbol{u} dQ = 0. \tag{2.3}$$

All the terms are integrated by parts except for the external force and enforce the essential (i.e. strong Dirichlet) and natural boundary conditions over $(P_n)_{\mathrm{g}}$ and $(P_n)_{\mathrm{h}}$, the complementary subsets of P_n. That gives us the following variational formulation: find $\boldsymbol{u} \in \mathcal{S}_{\boldsymbol{u}}$ and $p \in \mathcal{S}_p$ such that $\forall\ \mathbf{w} \in \mathcal{V}_{\boldsymbol{u}}$ and $\forall\ q \in \mathcal{V}_p$

$$\int_{\Omega_{n+1}} \mathbf{w}_{n+1}^- \cdot \rho \boldsymbol{u}_{n+1}^- d\Omega - \int_{\Omega_n} \mathbf{w}_n^+ \cdot \rho \boldsymbol{u}_n^- d\Omega$$
$$- \int_{Q_n} \frac{\partial \mathbf{w}}{\partial t} \cdot \rho \boldsymbol{u} dQ - \int_{(P_n)_{\mathrm{h}}} (\mathbf{w} \cdot \rho \boldsymbol{u}) (\mathbf{n} \cdot \mathbf{v}) \, dP$$
$$+ \int_{(P_n)_{\mathrm{h}}} (\mathbf{w} \cdot \rho \boldsymbol{u}) (\mathbf{n} \cdot \boldsymbol{u}) \, dP - \int_{Q_n} \boldsymbol{\nabla} \mathbf{w} : \rho \boldsymbol{u}\boldsymbol{u} dQ$$
$$- \int_{Q_n} \mathbf{w} \cdot \rho \mathbf{f} dQ - \int_{(P_n)_{\mathrm{h}}} \mathbf{w} \cdot \mathbf{h} dP + \int_{Q_n} \boldsymbol{\varepsilon}(\mathbf{w}) : \boldsymbol{\sigma} dQ$$
$$+ \int_{P_n} q \mathbf{n} \cdot \boldsymbol{u} dP - \int_{Q_n} \boldsymbol{\nabla} q \cdot \boldsymbol{u} dQ = 0, \tag{2.4}$$

where the notation $(\cdot)_n^-$ and $(\cdot)_n^+$ denotes the values at t_n as approached from below and above, and $\mathbf{v} = \frac{d\mathbf{x}}{dt}$ is the velocity of the spatial-domain boundary.

2.2.2 Scale separation

In the variational multiscale techniques [22, 25, 26, 7] the "coarse-scale" and "fine-scale" are separated as follows:

$$\mathcal{S}_{\boldsymbol{u}} = \overline{\mathcal{S}_{\boldsymbol{u}}} \oplus \mathcal{S}'_{\boldsymbol{u}}, \tag{2.5}$$

$$\mathcal{S}_p = \overline{\mathcal{S}_p} \oplus \mathcal{S}'_p, \tag{2.6}$$

$$\mathcal{V}_{\boldsymbol{u}} = \overline{\mathcal{V}_{\boldsymbol{u}}} \oplus \mathcal{V}'_{\boldsymbol{u}}, \tag{2.7}$$

$$\mathcal{V}_p = \overline{\mathcal{V}_p} \oplus \mathcal{V}'_p. \tag{2.8}$$

The coarse-scale part of Eq. (2.4) is written as follows:

$$\begin{aligned}
&\int_{\Omega_{n+1}} \overline{\mathbf{w}}^-_{n+1} \cdot \rho \boldsymbol{u}^-_{n+1} d\Omega - \int_{\Omega_n} \overline{\mathbf{w}}^+_n \cdot \rho \boldsymbol{u}^-_n d\Omega \\
&- \int_{Q_n} \frac{\partial \overline{\mathbf{w}}}{\partial t} \cdot \rho \boldsymbol{u} dQ - \int_{(P_n)_{\mathrm{h}}} \left(\overline{\mathbf{w}} \cdot \rho \boldsymbol{u}\right) \left(\mathbf{n} \cdot \mathbf{v}\right) dP \\
&+ \int_{(P_n)_{\mathrm{h}}} \left(\overline{\mathbf{w}} \cdot \rho \boldsymbol{u}\right) \left(\mathbf{n} \cdot \boldsymbol{u}\right) dP - \int_{Q_n} \boldsymbol{\nabla} \overline{\mathbf{w}} : \rho \boldsymbol{u} \boldsymbol{u} dQ \\
&- \int_{Q_n} \overline{\mathbf{w}} \cdot \rho \mathbf{f} dQ - \int_{(P_n)_{\mathrm{h}}} \overline{\mathbf{w}} \cdot \mathbf{h} dP + \int_{Q_n} \boldsymbol{\varepsilon}(\overline{\mathbf{w}}) : \boldsymbol{\sigma} dQ \\
&+ \int_{P_n} \overline{q} \mathbf{n} \cdot \boldsymbol{u} dP - \int_{Q_n} \boldsymbol{\nabla} \overline{q} \cdot \boldsymbol{u} dQ = 0.
\end{aligned} \tag{2.9}$$

From [22, 25, 26, 7], the fine-scale solutions are represented by the strong-form residuals of the coarse-scale:

$$\boldsymbol{u}' = -\frac{\tau_{\mathrm{M}}}{\rho} \mathbf{r}_{\mathrm{M}} \left(\overline{\boldsymbol{u}}, \overline{p}\right), \tag{2.10}$$

$$p' = -\rho \nu_{\mathrm{C}} r_{\mathrm{C}} \left(\overline{\boldsymbol{u}}\right), \tag{2.11}$$

where

$$\begin{aligned} \mathbf{r}_{\mathrm{M}}(\boldsymbol{u},p) &= \rho\left(\frac{\partial \boldsymbol{u}}{\partial t}+\boldsymbol{u}\cdot\boldsymbol{\nabla}\boldsymbol{u}-\mathbf{f}\right)+\boldsymbol{\nabla}p \\ &\quad -2\boldsymbol{\nabla}\cdot\mu\boldsymbol{\varepsilon}(\boldsymbol{u}), \qquad (2.12) \\ r_{\mathrm{C}}(\boldsymbol{u}) &= \boldsymbol{\nabla}\cdot\boldsymbol{u}, \qquad (2.13) \end{aligned}$$

and τ_{M} and ν_{C} are stabilization parameters measured in units of time and kinematic viscosity, respectively.

Remark 1 *More on the fine-scale approximation in conjunction with the Green's operator can be found in [22, 25, 26, 7].*

2.2.3 VMS DSD/SST formulation

As previously noted, in the DSD/SST method [53, 57, 58, 54, 62], the space–time finite element interpolation functions are continuous within a space–time slab, but discontinuous from one space–time slab to another. The finite-dimensional trial and test functions spaces for the velocity and pressure are denoted as $\boldsymbol{u}^h \in \left(\mathcal{S}_{\boldsymbol{u}}^h\right)_n$, $p^h \in \left(\mathcal{S}_p^h\right)_n$, $\mathbf{w}^h \in \left(\mathcal{V}_{\boldsymbol{u}}^h\right)_n$ and $q^h \in \left(\mathcal{V}_p^h\right)_n$.

Fine-scale discretization

The fine-scale solutions are evaluated over each element from Eqs. (2.10) and (2.11) with $\boldsymbol{u}^h \in \left(\mathcal{S}_{\boldsymbol{u}}^h\right)_n$ and $p^h \in \left(\mathcal{S}_p^h\right)_n$:

$$\begin{aligned} \boldsymbol{u}' &= -\frac{\tau_{\mathrm{M}}}{\rho}\mathbf{r}_{\mathrm{M}}\left(\boldsymbol{u}^h,p^h\right), \qquad (2.14) \\ p' &= -\rho\nu_{\mathrm{C}}r_{\mathrm{C}}\left(\boldsymbol{u}^h\right). \qquad (2.15) \end{aligned}$$

Remark 2 *When the polynomial order of the shape functions is less than two, the last term in Eq. (2.12) vanishes.*

There are various ways of defining τ_{M} and ν_{C}. For τ_{M} in this thesis, the following definition is used:

$$\tau_{\mathrm{M}} = \tau_{\mathrm{SUPG}}, \tag{2.16}$$

where τ_{SUPG} comes from [54], specifically the definition given by Eqs. (107)–(109) in [54], which can also be found as the definition given by Eqs. (7)–(9) in [62]. For ν_{C}, the ν_{LSIC} definition is considered as given in [62]:

$$\nu_{\mathrm{C}} = \nu_{\mathrm{LSIC}} = \tau_{\mathrm{SUPG}} \left\| \boldsymbol{u}^h - \mathbf{v}^h \right\|^2, \tag{2.17}$$

where $\mathbf{v}^h$ is the mesh velocity, and the definition from [6]:

$$\nu_{\mathrm{C}} = \left(\tau_{\mathrm{M}} \sum_{i=1}^{n_{sd}} G_{ii} \right)^{-1}, \tag{2.18}$$

where

$$G_{ij} = \sum_{k=1}^{n_{sd}} \frac{\partial \xi_k}{\partial x_i} \frac{\partial \xi_k}{\partial x_j}, \tag{2.19}$$

and ξ is the vector of element coordinates. In our computations the stabilization parameters are evaluated at $\xi = \mathbf{0}$. For more ways of calculating τ_{SUPG}, see [60, 2, 54, 55, 3, 56, 38, 15, 16, 21, 17].

Remark 3 *The τ_{SUGN12} component of the τ_{SUPG} definition given by Eqs. (107)–(109) in [54] is the space–time version of the original definition in [61]. These definitions sense the order of the interpolation functions in addition to the element geometry. While some τ definitions do this, others do not. The definitions in Sections 3.3.1 and 3.3.2 of [40], for example, are among those that do not.*

Remark 4 *Remark 3 is also applicable when the interpolation functions are NURBS functions. This includes classical p-refinement and also k-refinement, except when used in conjunction with periodic B-splines.*

Remark 5 *In meshes made of NURBS, for quadrilateral (or hexahedral) elements that degenerate to triangles (or tetrahedra), we calculate τ_{SUGN12}, τ_{SUGN1} when applicable, and "h_{RGN}" embedded in the τ_{SUGN3} definition in a special way. Instead of letting the sum of the magnitudes involved in the expression degenerate, we first add together the basis functions associated with the coalescing control points, and then apply the expression using the modified basis functions. In other words, we do not degenerate the expression, but instead apply the expression to the degenerated basis functions. This special way is applicable also in the context of finite element meshes.*

Coarse-scale discretization

Spatially discretized version of Eq. (2.9) is written as follows: find $\boldsymbol{u}^h \in \left(\mathcal{S}_{\boldsymbol{u}}^h\right)_n$ and $p^h \in \left(\mathcal{S}_p^h\right)_n$ such that $\forall\ \mathbf{w}^h \in \left(\mathcal{V}_{\boldsymbol{u}}^h\right)_n$ and $\forall\ q^h \in \left(\mathcal{V}_p^h\right)_n$:

$$
\begin{aligned}
&\int_{\Omega_{n+1}} (\mathbf{w}^h)_{n+1}^- \cdot \rho \left((\boldsymbol{u}^h)_{n+1}^- + (\boldsymbol{u}')_{n+1}^- \right) d\Omega \\
&- \int_{\Omega_n} (\mathbf{w}^h)_n^+ \cdot \rho \left((\boldsymbol{u}^h)_n^- + (\boldsymbol{u}')_n^- \right) d\Omega \\
&- \int_{Q_n} \frac{\partial \mathbf{w}^h}{\partial t} \cdot \rho (\boldsymbol{u}^h + \boldsymbol{u}') dQ \\
&+ \int_{(P_n)_{\mathrm{h}}} \left(\mathbf{w}^h \cdot \rho \left(\boldsymbol{u}^h + \boldsymbol{u}' \right) \right) \left(\mathbf{n}^h \cdot (\boldsymbol{u}^h + \boldsymbol{u}' - \mathbf{v}^h) \right) dP \\
&- \int_{Q_n} \boldsymbol{\nabla} \mathbf{w}^h : \rho (\boldsymbol{u}^h + \boldsymbol{u}') (\boldsymbol{u}^h + \boldsymbol{u}') dQ - \int_{Q_n} \mathbf{w}^h \cdot \rho \mathbf{f}^h dQ \\
&- \int_{(P_n)_{\mathrm{h}}} \mathbf{w}^h \cdot \mathbf{h}^h dP + \int_{Q_n} \boldsymbol{\varepsilon}(\mathbf{w}^h) : \left(\boldsymbol{\sigma}(p^h, \boldsymbol{u}^h) + \boldsymbol{\sigma}' \right) dQ \\
&+ \int_{P_n} q^h \mathbf{n}^h \cdot (\boldsymbol{u}^h + \boldsymbol{u}') dP \\
&- \int_{Q_n} \boldsymbol{\nabla} q^h \cdot (\boldsymbol{u}^h + \boldsymbol{u}') dQ = 0.
\end{aligned}
\tag{2.20}
$$

Here $\boldsymbol{\sigma}' \equiv \boldsymbol{\sigma} - \boldsymbol{\sigma}^h$ is introduced temporarily. The fine-scale solution is set to zero at the spatial and temporal boundaries, and then the assumption $\boldsymbol{\varepsilon}(\mathbf{w}^h) : 2\mu \boldsymbol{\nabla} \boldsymbol{u}' = 0$

(see [26, 24]) is applied to obtain the following form:

$$
\begin{aligned}
&\int_{\Omega_{n+1}} (\mathbf{w}^h)^-_{n+1} \cdot \rho (\boldsymbol{u}^h)^-_{n+1} d\Omega - \int_{\Omega_n} (\mathbf{w}^h)^+_n \cdot \rho (\boldsymbol{u}^h)^-_n d\Omega \\
&- \int_{Q_n} \frac{\partial \mathbf{w}^h}{\partial t} \cdot \rho (\boldsymbol{u}^h + \boldsymbol{u}') dQ \\
&+ \int_{(P_n)_{\mathrm{h}}} \left(\mathbf{w}^h \cdot \rho \boldsymbol{u}^h \right) \left(\mathbf{n}^h \cdot (\boldsymbol{u}^h - \mathbf{v}^h) \right) dP \\
&- \int_{Q_n} \boldsymbol{\nabla} \mathbf{w}^h : \rho (\boldsymbol{u}^h + \boldsymbol{u}') (\boldsymbol{u}^h + \boldsymbol{u}') dQ - \int_{Q_n} \mathbf{w}^h \cdot \rho \mathbf{f}^h dQ \\
&- \int_{(P_n)_{\mathrm{h}}} \mathbf{w}^h \cdot \mathbf{h}^h dP + \int_{Q_n} \boldsymbol{\varepsilon}(\mathbf{w}^h) : \boldsymbol{\sigma}(p^h + p', \boldsymbol{u}^h) dQ \\
&+ \int_{P_n} q^h \mathbf{n}^h \cdot \boldsymbol{u}^h dP - \int_{Q_n} \boldsymbol{\nabla} q^h \cdot (\boldsymbol{u}^h + \boldsymbol{u}') dQ = 0.
\end{aligned} \tag{2.21}
$$

Comparison with the original DSD/SST formulation

The terms in the formulation given by Eq. (2.21) can be further rearranged to compare it with the original DSD/SST formulation (with the advection term retained in the conservation-law form) and obtain the following:

$$
\begin{aligned}
&\int_{Q_n} \mathbf{w}^h \cdot \rho \left(\frac{\partial \boldsymbol{u}^h}{\partial t} + \boldsymbol{\nabla} \cdot (\boldsymbol{u}^h \boldsymbol{u}^h) - \mathbf{f}^h \right) dQ \\
&+ \int_{Q_n} \boldsymbol{\varepsilon}(\mathbf{w}^h) : \boldsymbol{\sigma}(p^h, \boldsymbol{u}^h) dQ \\
&- \int_{(P_n)_{\mathrm{h}}} \mathbf{w}^h \cdot \mathbf{h}^h dP + \int_{Q_n} q^h \cdot \boldsymbol{\nabla} \boldsymbol{u}^h dQ \\
&+ \int_{\Omega_n} (\mathbf{w}^h)^+_n \cdot \rho \left((\boldsymbol{u}^h)^+_n - (\boldsymbol{u}^h)^-_n \right) d\Omega \\
&- \sum_{e=1}^{(n_{el})_n} \int_{Q^e_n} \left[\rho \left(\frac{\partial \mathbf{w}^h}{\partial t} + \boldsymbol{u}^h \cdot \boldsymbol{\nabla} \mathbf{w}^h \right) + \boldsymbol{\nabla} q^h \right] \cdot \boldsymbol{u}' dQ \\
&- \sum_{e=1}^{(n_{el})_n} \int_{Q^e_n} \boldsymbol{\nabla} \cdot \mathbf{w}^h p' dQ - \sum_{e=1}^{(n_{el})_n} \int_{Q^e_n} \rho \boldsymbol{u}' \cdot \left(\boldsymbol{\nabla} \mathbf{w}^h \right) \boldsymbol{u}^h dQ \\
&- \sum_{e=1}^{(n_{el})_n} \int_{Q^e_n} \rho \boldsymbol{u}' \cdot \left(\boldsymbol{\nabla} \mathbf{w}^h \right) \boldsymbol{u}' dQ = 0.
\end{aligned} \tag{2.22}
$$

Here each Q_n is decomposed into elements Q_n^e, where $e = 1, 2, \ldots, (n_{el})_n$. The subscript n used with n_{el} is for the general case where the number of space–time elements may change from one space–time slab to another.

Remark 6 *The last two terms correspond to the Reynolds stress and cross-stress, respectively. This formulation is called DSD/SST-VMST (i.e. the version with the variational multiscale turbulence model).*

Remark 7 *If the last two terms are excluded, the formulation is the same as the original DSD/SST formulation (with the advection term retained in the conservation-law form) under the conditions that $\tau_{PSPG} = \tau_{SUPG}$ and $\nu_C = \nu_{LSIC}$. The 6th and 7th terms are the SUPG/PSPG and LSIC (least-squares on incompressibility constraint) stabilization terms, respectively. This is named DSD/SST-SUPS (i.e. the version with the SUPG/PSPG stabilization).*

Remark 8 *One of the main differences between the ALE and DSD/SST forms of the variational multiscale method is that the DSD/SST formulation retains the fine-scale time derivative term $\frac{\partial \boldsymbol{u}'}{\partial t}\big|_{\xi}$. Dropping this term is called the "quasi-static" assumption (see [10] for the terminology). This is the same as the WTSE option in the DSD/SST formulation (see Remark 2 of [62]). It is believed that this makes a significant difference, especially when the polynomial orders in space or time are higher (see Section 6 of [49]).*

2.3 New Generation Space–Time Formulations

The flapping flight computations in this thesis utilize the new generation space–time formulations, namely the formulations' extension to NURBS. Thus, from the DSD/SST-VMST formulation, it is useful to show the extension of these formulations. The information in this section is nearly the same as it is presented in [49]. For more

information, such as the results of stability and accuracy analysis, the interested reader should see this reference.

New generation DSD/SST formulations, with emphasis on increasing the robustness and lowering the computational cost, were introduced by the T★AFSM in [62]. The new versions were named "DSD/SST-SP", "DSD/SST-TIP1" and "DSD/SST-SV" to differentiate them from the original version introduced in [53, 57, 58], which was named "DSD/SST-DP" in [62]. In this section, additions to the new-generation space–time formulations developed in [62] are introduced, mainly the extension to the space–time formulations with NURBS, and the original DSD/SST formulation (i.e. DSD/SST-DP) is redeployed. A space–time basis function can be written as a product of its spatial and temporal parts:

$$N_a^\alpha = T^\alpha(\theta)\, N_a(\xi), \qquad \begin{aligned} a &= 1, 2, \ldots, n_{en}, \\ \alpha &= 1, 2, \ldots, n_{ent}, \end{aligned} \tag{2.23}$$

where $\theta \in [-1,\ 1]$ is the temporal element coordinate, and n_{en} and n_{ent} are the number of spatial and temporal element nodes (see Figure 2.1 for an example of temporal basis functions). In general, the values

$$\phi_n^- = \lim_{t \to t_n^-} \phi^h(t), \tag{2.24}$$

$$\phi_n^+ = \lim_{t \to t_n^+} \phi^h(t), \tag{2.25}$$

do not need to be equal to $\phi_{n-1}^{n_{ent}}$ and ϕ_n^1 (coefficients of the basis functions $T_{n-1}^{n_{ent}}$ and T_n^1). However, for the test cases considered in [49], the basis functions are interpolatory at $\theta = -1$ and $\theta = 1$ and therefore $\phi_n^- = \phi_{n-1}^{n_{ent}}$ and $\phi_n^+ = \phi_n^1$.

While all components (i.e. unknowns) and the corresponding test functions are discretized with the same set of spatial basis functions N_a, they may be discretized with different sets of temporal basis functions T_ζ^α, where ζ indicates the component.

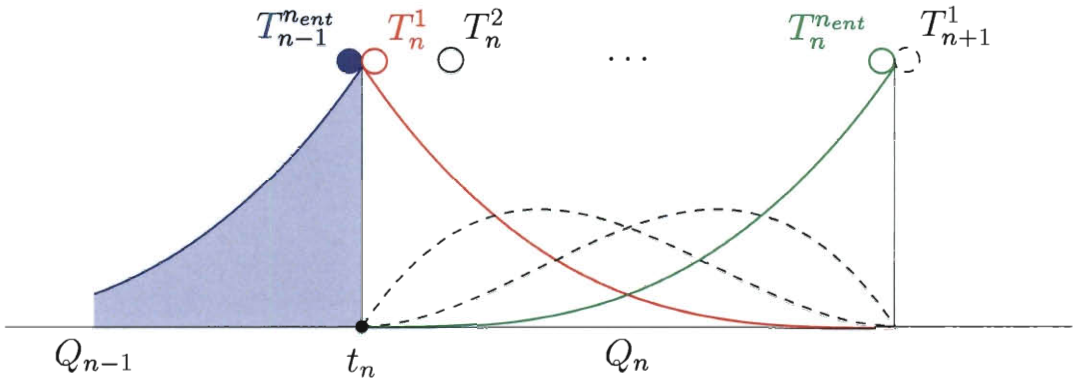


Figure 2.1: Example of temporal basis functions.

Another mapping, $\Theta_\zeta(\theta) \in [-1,\ 1]$, is introduced for representing the temporal functions, where $\Theta_\zeta(\theta)$ is a strictly increasing function. The generalized space–time basis function for the element indices $(a,\ \alpha)$ can then be rewritten as follows:

$$\left(N_a^\alpha\right)_\zeta = T_\zeta^\alpha\left(\Theta_\zeta(\theta)\right) N_a\left(\xi\right). \tag{2.26}$$

For mesh motion, $\Theta\left(\theta\right) \neq \theta$ is allowed (see Section 5.1 of [49]). Prescribed and unknown variables can be represented over different space–time slabs because only there is only a need to supply the prescribed values at the integration points. Most of the time, higher-order basis functions can represent complex functions with a fewer number of control points. This is very helpful in decreasing the I/O intensity, such as in computations with the multiscale SCFSI techniques (see Section 3 of [49]).

Chapter 3

Special Modeling Techniques

To more accurately compute flapping flight aerodynamics, various special modeling techniques have been developed to supplement the DSD/SST-VMST formulation presented in Chapter 2. These special modeling techniques leverage the benefits of higher-order functions, such as NURBS, and were first discussed in [43]. Section 3.1 describes a means of conducting temporal interpolation with NURBS, which is used in describing the prescribed motion of the locust. Section 3.2 provides a description of the SSDM, which can be used to convert experimental data into data suitable for prescribed motion in a computation. Sections 3.3 and 3.4 further explain the application of temporal NURBS in mesh update techniques and fluid mechanics computations, respectively. Below is a discussion of these techniques that has been slightly-modified from the version in [43].

Recently there has been an even better understanding and articulation [50, 49] of the desirable features of the DSD/SST formulation as a moving-mesh technique. For example, as pointed out in [50, 49], using higher-order basis functions for the temporal representation in a space–time computation gives us better solution accuracy, and would be essential in getting full benefit out of using higher-order, such as NURBS [23, 9, 8, 11], basis functions in space. As also pointed out partly in [49], in the space–time flow computations, using NURBS basis functions for the tempo-

ral representation of the motion and deformation of the solid surfaces and also for the motion and deformation of the volume meshes computed provides a better temporal representation of the solid surfaces and a more effective way of handling the volume-mesh motion.

3.1 Temporal Interpolation

3.1.1 Time Representation

As described in Section 5 in [49], NURBS basis functions can be used for both spatial and temporal discretization. The focus in this section is on the temporal part. Time $t \in (0, T)$ is represented with with p^{th} order NURBS basis functions. The p^{th} order NURBS basis functions R^{β} ($\beta = 0, \ \cdots, \ n_{\mathrm{c}} - 1$) are defined on the parametric space, which is defined by the open knot vector $\{\vartheta_1, \ \cdots, \ \vartheta_{n_{\mathrm{t}}}\}$, where n_{c} and n_{t} are the number of control points and knots. Then, time t is represented as follows:

$$t = \sum_{\beta=0}^{n_{\mathrm{c}}-1} t_c^{\beta} R^{\beta}(\vartheta), \tag{3.1}$$

where t_c^{β} represents the temporal-control point. In the case of the space–time formulation there is a mesh corresponding to each temporal-control point t_c^{β}. Elements in the time direction and the element coordinate $\theta \in [-1, 1]$ are defined; another mapping $\Theta(\theta) \in [-1, 1]$ is introduced with a strictly increasing function, which is also described in [49]. This element coordinate and the NURBS parametric space are related as follows:

$$\vartheta = \frac{(1 - \Theta(\theta))\vartheta_{e+p+1} + (1 + \Theta(\theta))\vartheta_{e+p+2}}{2}, \tag{3.2}$$

where e represents the element index ($e = 0, \cdots n_{\mathrm{e}} - 1$, where n_{e} is the number of elements). It is assumed that there is no knot multiplicity inside the knot vector, and

the following condition is satisfied: $n_e = n_t - 2p - 1$ and $n_c = n_t - p - 1$. The element local shape functions are defined as follows:

$$T_e^{\alpha}(\Theta(\theta)) = R^{e+\alpha-1}(\vartheta(\theta)). \tag{3.3}$$

In the time interval of element e, t can be represented with the local shape functions as follows:

$$t(\Theta(\theta)) = \sum_{\alpha=1}^{p+1} t_c^{e+\alpha-1} T_e^{\alpha}(\Theta(\theta)). \tag{3.4}$$

Remark 9 *In the discussion above, θ is used as the element coordinate within the corresponding temporal element; i.e. the coordinate for the numerical integration. The function $\Theta(\theta)$ re-parametrizes the element coordinate. This adds flexibility to temporal representation, which is attractive in some cases. For example, an arc can be represented by NURBS, however, in this case, a constant speed cannot be represented on the arc. The re-parametrization allows for a constant speed on the arc (see Section 3.1.5).*

3.1.2 Time Marching Problem

Consider a set of NURBS basis functions that is being used in representing data that we will work with. Figure 3.1 shows an example. Starting from this, we can form a new basis set by knot insertion with the objective that all elements become patches as shown in Figure 3.2. After that, we can use this basis set in our space–time computation while representing exactly the data.

Instead, we propose an alternative process that will have the same functionality, but without the need to explicitly represent the data we are working with using the new basis set. In that process, we simply form a new basis set where each element is a patch, and the data is represented in our formulation in terms of its own basis set. In general, the basis set that we simply form does not need to be the same as

the one that could be obtained by the process described earlier, but if it is, then the two processes result in equivalent solution methods. Figure 3.3 shows the new basis functions that we simply form.

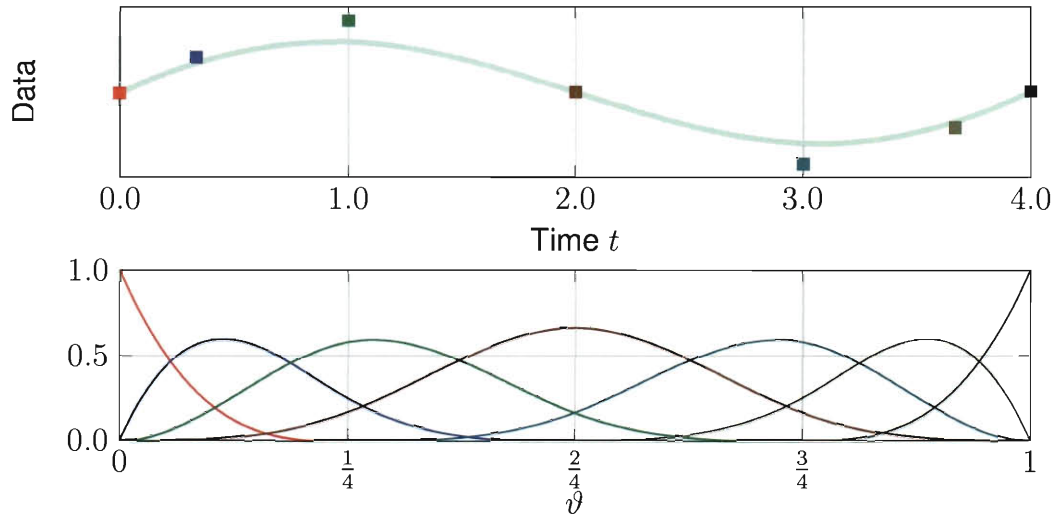


Figure 3.1: Data represented with NURBS with the data and control variables (top) and the basis functions corresponding to each of the control variables (bottom).

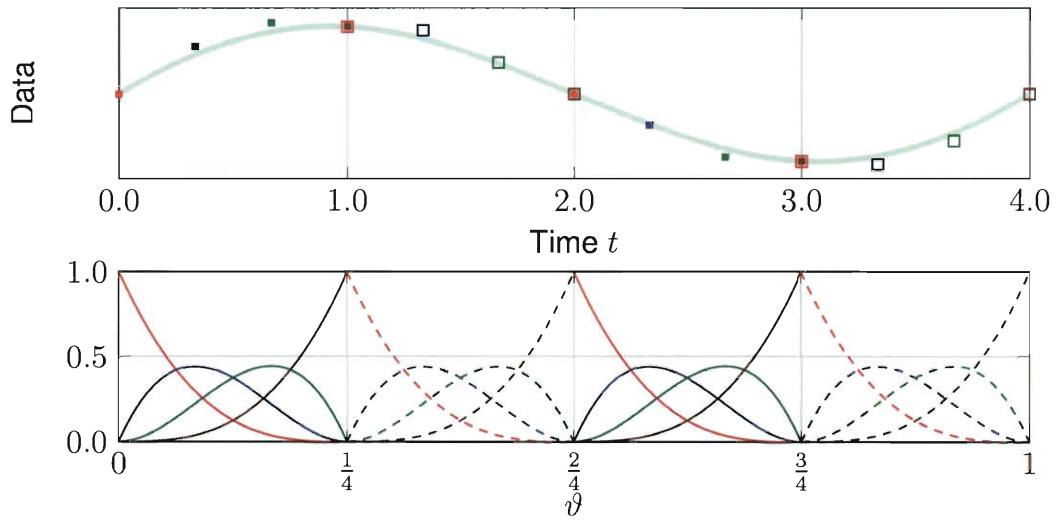


Figure 3.2: Data represented with basis functions after knot insertion with data and control variables (top) and the basis functions corresponding to each of the control variables (bottom).

Since we need to work with different basis sets, we map one parametric space to

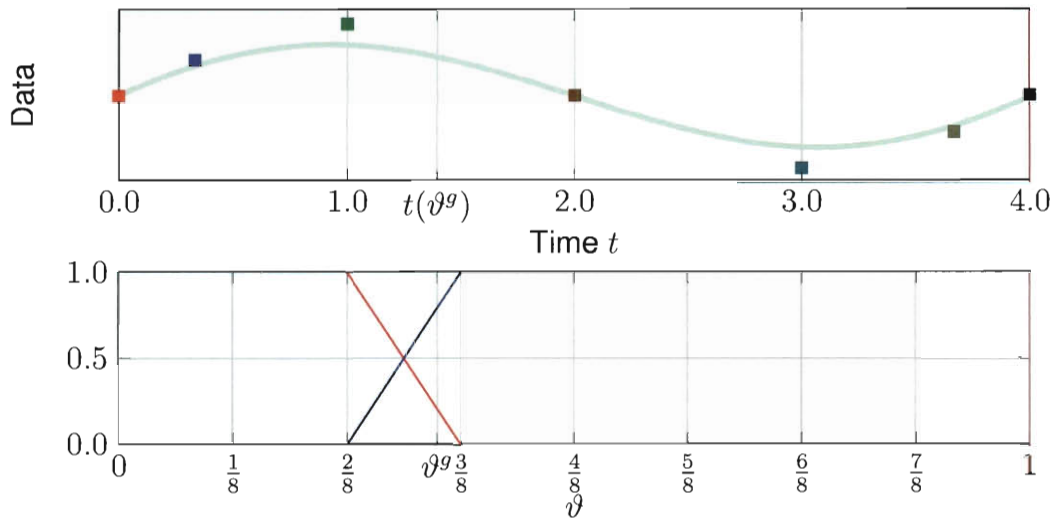


Figure 3.3: The data and control variables (top) with the simply-formed basis functions for a given interval for the space–time computation (bottom). To integrate over the interval in the NURBS representation of the data, it is necessary to search for the corresponding element and parametric coordinate for the time $t(\vartheta^g)$ of each quadrature point ϑ^g, and interpolate the value from the data.

the other through physical time t. With the function defined by Eq. (3.1), time t can be obtained from the parametric space ϑ. Here we consider the inverse functionality; i.e., $t \to \vartheta$. We find the parametric space coordinate as follows:

1. Find the element e that is represented by the knot span $(\vartheta_{e+p+1}, \vartheta_{e+p+2})$. The process requires only time values at each element boundary, and the element index e can be quickly obtained by using a binary search technique.

2. Solve θ for a given t by using Newton–Raphson iterations as follows:

$$\theta^{i+1} = \theta^i - \left(t - t(\theta^i)\right) \left(\left. \frac{\mathrm{d}t}{\mathrm{d}\theta} \right|^i \right)^{-1} \tag{3.5}$$

where superscript "i" is the iteration counter, $t(\theta^i)$ can be calculated from Eq. (3.4), and

$$\left. \frac{\mathrm{d}t}{\mathrm{d}\theta} \right|^i = \sum_{\alpha=1}^{p+1} t_c^{e+\alpha-1} \left. \frac{\mathrm{d}T_e^\alpha}{\mathrm{d}\Theta} \right|_{\Theta(\theta^i)} \left. \frac{\mathrm{d}\Theta}{\mathrm{d}\theta} \right|_{\theta^i} . \tag{3.6}$$

For the initial guess, $\theta^0 = 0$ is used.

3. Compute ϑ from Eq. (3.2).

3.1.3 Design of Temporal NURBS Basis Functions

The previous section described how to find the parametric space value corresponding to physical time. In this section, some specific temporal representations are described.

The time interval of the space–time slab is restricted for implementation convenience and computational efficiency such that the time interval does not step over a time corresponding to a temporal knot. Thus, the supporting set of meshes during the time integration consists of only specific $p+1$ meshes. Because of this requirement, a uniform element size, i.e. $t(\vartheta_{e+p+2}) - t(\vartheta_{e+p+1}) = \Delta t$, where $\Delta t = \frac{T}{n_e}$, is convenient. Moreover, the following requirement may also be imposed:

$$\frac{\mathrm{d}t}{\mathrm{d}\theta} = \frac{\Delta t}{2}. \tag{3.7}$$

In the case of B-spline basis functions with the identical mapping $\Theta(\theta) = \theta$, the condition expressed by Eq. (3.7) can be obtained by selecting the control points as follows:

$$t_c^\beta = t_c^{\beta-1} + \frac{\vartheta_{\beta+p+1} - \vartheta_{\beta+1}}{p\,(\vartheta_{n_t} - \vartheta_1)} T, \tag{3.8}$$

for $\beta = 1, \ \cdots, \ n_c - 1$ and $t_c^0 = 0$.

3.1.4 Approximation in Time

Let $\mathbf{x}_A^s$ be the sampling values of a time-varying spatial position vector $\mathbf{x}_A$ at sampling times t^s ($s = 0, \ \cdots, \ n_{sp} - 1$, where n_{sp} is the number of sampling points). For example, $\mathbf{x}_A$ could be the position vector for spatial node A, or it could be the position vector for a point on a surface geometry extracted from video data. For each A, we want to represent the path corresponding to the sampling points with NURBS.

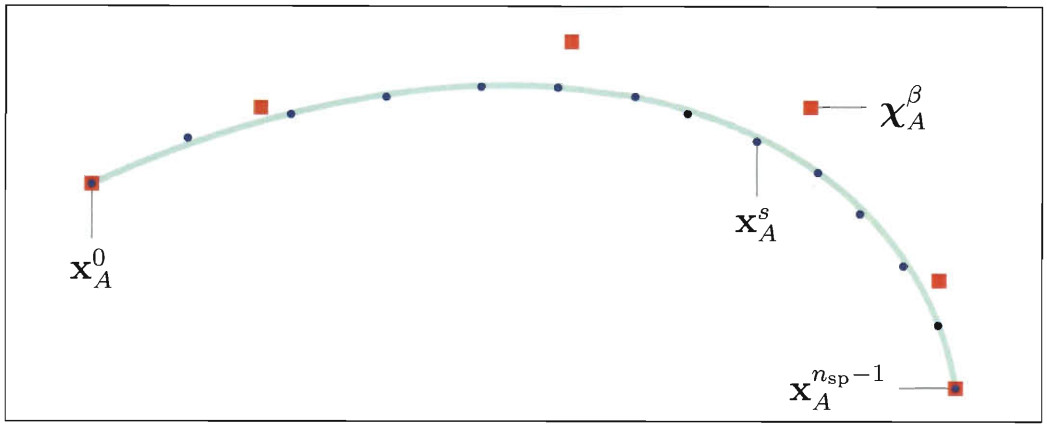


Figure 3.4: NURBS representation for a time-varying spatial position vector. The circles are the spatial position vector at each sampling time. The squares are the temporal-control points, which represent the smooth curve.

This serves two purposes: bringing smoothness to the temporal representation, and better representation accuracy for less control points. First, a linear finite element mesh is formed consisting of two-node elements. Then, the linear finite element mesh is translated to a NURBS representation using the following least-squares projection:

$$\int_0^T \boldsymbol{R}_A^h \cdot \left(\boldsymbol{\chi}_A^h - \mathbf{x}_A^h\right) \mathrm{d}t = 0, \tag{3.9}$$

where $\boldsymbol{R}_A^h$ and $\boldsymbol{\chi}_A^h$ are the respective test function and NURBS representation in time and $\mathbf{x}_A^h$ is the linear representation in time. Thus, the control point $\boldsymbol{\chi}_A^\beta$ corresponding to each time control point t_c^β is obtained. Figure 3.4 is an example.

Remark 10 *This is a simple projection. However, the concept is applicable to more complicated formulations to obtain smoother motion.*

3.1.5 An Example: Circular-Arc Motion

Path Representation

NURBS temporal basis functions would be useful in representing a particle path on a circular arc. Let us select as the origin the center of the circle to which the arc

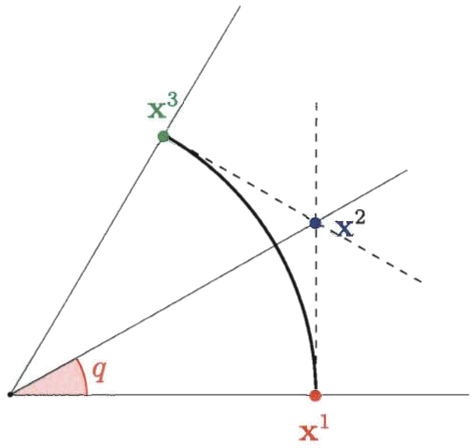


Figure 3.5: A circular arc represented by quadratic NURBS.

belongs. The particle travels from $\mathbf{x}^1$ to $\mathbf{x}^3$, $\|\mathbf{x}^1\| = \|\mathbf{x}^3\|$ as shown in Figure 3.5. It is known that a circular arc can be represented exactly by three control points with quadratic NURBS basis functions (it is valid only for $q < \frac{\pi}{2}$). The weights are $w_1 = w_3 = 1$, and $w_2 = \cos q$, where

$$\cos 2q = \frac{\mathbf{x}^1 \cdot \mathbf{x}^3}{r^2}, \tag{3.10}$$

$$r = \|\mathbf{x}^1\| = \|\mathbf{x}^3\|. \tag{3.11}$$

This results in the following temporal basis functions:

$$T^1(\Theta) = \frac{(1-\Theta)^2}{2\left((1+\Theta^2) + w_2\left(1-\Theta^2\right)\right)}, \tag{3.12}$$

$$T^2(\Theta) = \frac{w_2(1-\Theta^2)}{(1+\Theta^2) + w_2\left(1-\Theta^2\right)}, \tag{3.13}$$

$$T^3(\Theta) = \frac{(1+\Theta)^2}{2\left((1+\Theta^2) + w_2\left(1-\Theta^2\right)\right)}. \tag{3.14}$$

and the control points are $\mathbf{x}^1$,

$$\mathbf{x}^2 = \frac{r}{w_2}\frac{\mathbf{x}^1 + \mathbf{x}^3}{\|\mathbf{x}^1 + \mathbf{x}^3\|} \tag{3.15}$$

$$= \frac{1}{2w_2^2}\left(\mathbf{x}^1 + \mathbf{x}^3\right), \tag{3.16}$$

and $\mathbf{x}^3$. Thus, the arc can be represented as follows:

$$\mathbf{x}(\Theta) = \mathbf{x}^1 T^1(\Theta) + \mathbf{x}^2 T^2(\Theta) + \mathbf{x}^3 T^3(\Theta). \tag{3.17}$$

Constant Angular Velocity

First, by using Eq. (3.16), Eq. (3.17) is rearranged as follows:

$$\begin{aligned} \mathbf{x}(\Theta) &= \underbrace{\left(T^1(\Theta) + \frac{1}{2w_2^2} T^2(\Theta)\right)}_{Q^1(\Theta)} \mathbf{x}^1 \\ &\quad + \underbrace{\left(T^3(\Theta) + \frac{1}{2w_2^2} T^2(\Theta)\right)}_{Q^3(\Theta)} \mathbf{x}^3, \end{aligned} \tag{3.18}$$

where Q^1 and Q^3 are introduced for notation convenience. Taking the cross product with the unit vector along $\mathbf{x}^2$ results in

$$\frac{\mathbf{x}^1 + \mathbf{x}^3}{\|\mathbf{x}^1 + \mathbf{x}^3\|} \times \mathbf{x}(\Theta) = \frac{\mathbf{x}^1 \times \mathbf{x}^3}{r^2 \sin(2q)} r \sin(\omega t), \tag{3.19}$$

where $-\frac{\Delta t}{2} \leq t \leq \frac{\Delta t}{2}$ and $\omega \Delta t = 2q$ for notation convenience. From Eq. (3.19), this leads to

$$\frac{\mathbf{x}^1 + \mathbf{x}^3}{2r \cos q} \times \mathbf{x}(\Theta) = \frac{\mathbf{x}^1 \times \mathbf{x}^3}{2r \sin q \cos q} \sin(\omega t). \tag{3.20}$$

From Eqs. (3.18) and (3.20), we get

$$\left(Q^3 - Q^1\right) = \frac{\sin(\omega t)}{\sin q}. \tag{3.21}$$

From Eq. (3.21), it can be shown that

$$\Theta = \frac{\sin q}{1 - \cos q} \frac{\sin(\omega t)}{1 + \cos(\omega t)}. \tag{3.22}$$

Different mappings, $\Theta_{\mathbf{x}}$ and Θ_t, are used for the particle path and time. From Eq. (3.22), the mapping for the path is

$$\Theta_{\mathbf{x}} = \frac{\sin q}{1-\cos q}\frac{\sin(\omega t)}{1+\cos(\omega t)}. \tag{3.23}$$

It is assumed that time is represented with the same basis functions, and therefore

$$\begin{aligned} t(\Theta_t) &= \frac{\Delta t}{2}\left(T^3(\Theta_t) - T^1(\Theta_t)\right) \\ &= \frac{\Delta t \Theta_t}{1+\Theta_t^2+(1-\Theta_t^2)\cos q}, \end{aligned} \tag{3.24}$$

and,

$$\frac{\mathrm{d}t}{\mathrm{d}\theta} = \Delta t \frac{1-\Theta_t^2+(1+\Theta_t^2)\cos q}{\left(1+\Theta_t^2+(1-\Theta_t^2)\cos q\right)^2}\frac{\mathrm{d}\Theta_t}{\mathrm{d}\theta}. \tag{3.25}$$

- If $\Theta_t = \theta$ is selected:

  Equation (3.23) is evaluated by substituting Eq. (3.24). On the other hand, the derivative $\frac{\mathrm{d}t}{\mathrm{d}\theta}$ becomes

  $$\frac{\mathrm{d}t}{\mathrm{d}\theta} = \Delta t \frac{1-\theta^2+(1+\theta^2)\cos q}{\left(1+\theta^2+(1-\theta^2)\cos q\right)^2}, \tag{3.26}$$

  and

  $$\begin{aligned} \frac{\mathrm{d}\Theta_{\mathbf{x}}}{\mathrm{d}\theta} &= \frac{2q\sin q}{1-\cos q}\frac{1+\cos(\omega t)}{\left(1+\cos(\omega t)\right)^2} \\ &\quad \times \frac{1-\theta^2+(1+\theta^2)\cos q}{\left(1+\theta^2+(1-\theta^2)\cos q\right)^2}. \end{aligned} \tag{3.27}$$

- If $\frac{\mathrm{d}t}{\mathrm{d}\theta} = \frac{\Delta t}{2}$ is selected:

  From Eq. (3.25):

  $$\frac{\mathrm{d}\Theta_t}{\mathrm{d}\theta} = \frac{1}{2}\frac{\left(1+\Theta_t^2+(1-\Theta_t^2)\cos q\right)^2}{1-\Theta_t^2+(1+\Theta_t^2)\cos q}. \tag{3.28}$$

The mapping for the particle path becomes:

$$\Theta_{\mathbf{x}} = \frac{\sin q}{1 - \cos q} \frac{\sin(q\theta)}{1 + \cos(q\theta)}, \tag{3.29}$$

and

$$\frac{\mathrm{d}\Theta_{\mathbf{x}}}{\mathrm{d}\theta} = \frac{q \sin q}{1 - \cos q} \frac{1}{1 + \cos(q\theta)}. \tag{3.30}$$

3.2 SSDM

Suppose we want to track the motion/deformation of an object with surface shape that is too complex to track in full detail, giving us just the option of tracking only a finite number of points belonging to this complex shape. In the simple-shape deformation model (SSDM), it is assumed that those tracked points are associated with a simple shape (SS) instead of the actual, complex shape. NURBS will be used for the spatial representation of the SS, and we note that the SS will be larger than the complex shape.

Starting with the reference configuration, the SS, the complex shape and the tracked points all can be seen in a common parametric space (Figure 3.6). The

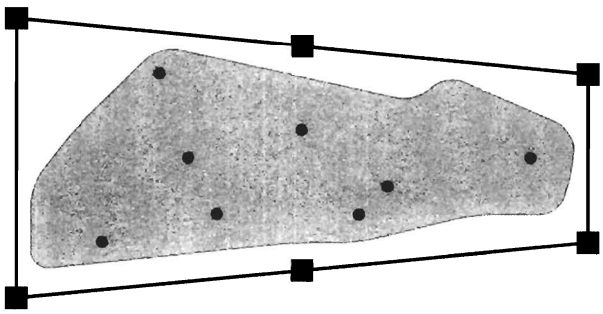


Figure 3.6: Example of SSDM. Complex shape is shaded. Circles are tracked points. SS is represented by squares (control points).

complex shape can be represented by finite elements or NURBS. Control points of the SS at different times during tracking are determined by a least-squares fit. The

fit minimizes the difference between the positions on the SS (with respect to the reference configuration) of the tracked points and the positions of the actual tracked points. The complex shape at a given time is determined by interpolation from the parametric space in the case of the finite element representation, and by least-square projection in the case of NURBS representation. The least-squares integration is over the parametric space of the complex shape, and the difference between the complex-shape and simple shape representations is minimized with respect to the control points of the complex shape. In the full space–time representation, the method described above is applied to temporal-control values that are determined as described in Section 3.1.4 instead of the actual physical locations.

3.3 Mesh Update Technique

3.3.1 Mesh Computation and Representation

Given the surface mesh, the volume mesh is computed using the mesh moving technique described in [51]. Here this technique is applied to computing the meshes that will serve as temporal control points. This allows for a longer time in between mesh computations, but we get the meshes from the temporal representation whenever and at whatever frequency is required. Obviously, this also reduces the storage amount and access associated with the meshes. However, because of the longer time between the control meshes, sub-iterations might be needed in computing those meshes with the mesh moving technique mentioned.

Remark 11 *We note that obtaining the meshes used in the computations from the temporal representation can be done independent of which time direction was used in computing the control meshes.*

3.3.2 Remeshing Technique

In many computations, remeshing becomes unavoidable. When it is necessary to remesh, there are two choices. To explain those two choices, let us assume that when trying to move from control mesh M_c^β to $M_c^{\beta+1}$, it is found that the quality of $M_c^{\beta+1}$ is less than desirable. In the first choice, which is called "trimming", we remesh going back to $M_c^{\beta-p+1}$. Then, whenever our solution process needs a mesh, depending on the time, the control meshes belongs to either only the un-remeshed set or only the re-meshed set (Figure 3.7) are used.

In the second choice, knot insertion is performed p times in the temporal representation of the surface at the right-most knot before the maximum value of the basis function corresponding to $t_c^{\beta+1}$, making that knot a new patch boundary. Then we do the mesh moving computation for the control meshes associated with the newly-defined basis functions— not only the one at the new patch boundary, but also going back $(p-1)$ basis functions (Figure 3.8).

The second choice is used in these computations because we believe that in many cases the need for remeshing is generated by a topological change, which can be avoided going over with a large step if the knot insertion process is used.

3.4 Fluid Mechanics Computation with Temporal NURBS Mesh

We solve the fluid dynamics equations with the DSD/SST formulation. Here we explain two techniques related to the moving-mesh problem.

3.4.1 No-Slip Condition on a Prescribed Boundary

Suppose we have a prescribed mesh motion, and no-slip conditions on part of the boundary of that mesh. Those Dirichlet conditions can be obtained from the mesh

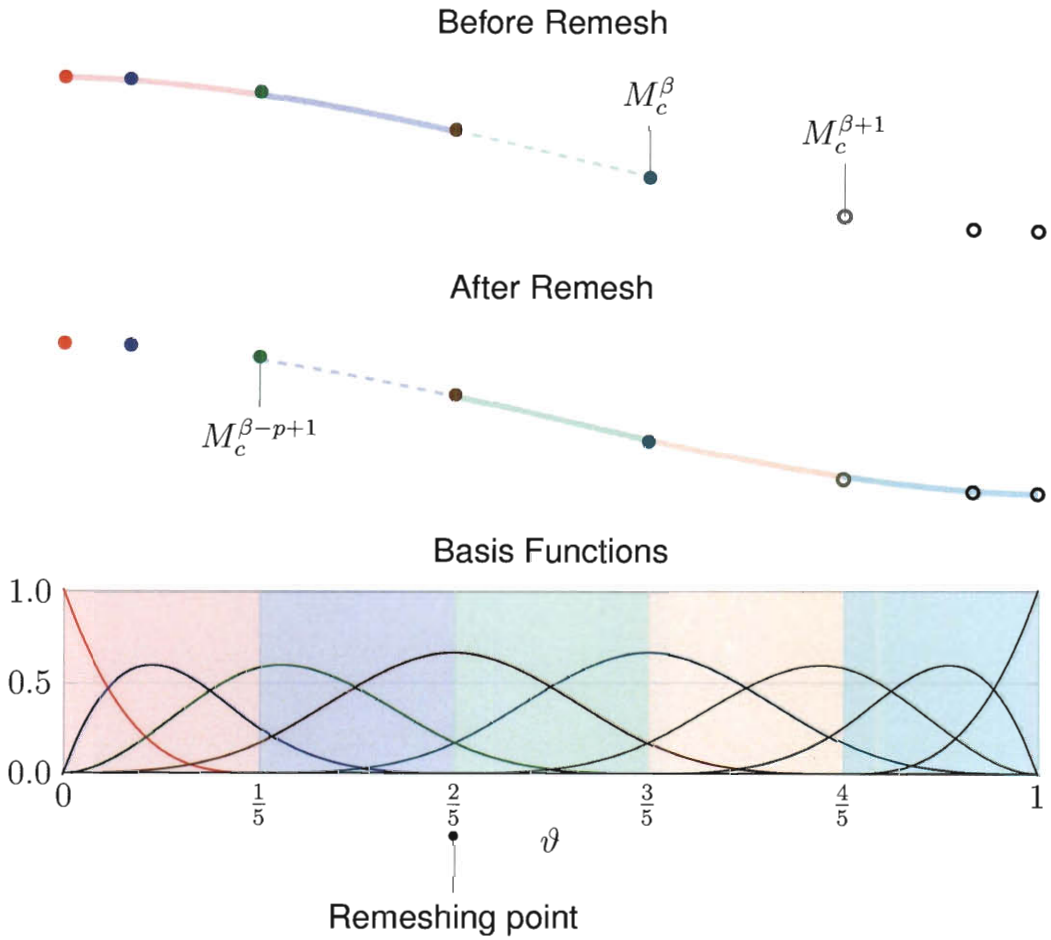


Figure 3.7: Remeshing and trimming NURBS. A set of un-remeshed meshes (top). A set of remeshed meshes (middle). Common basis functions (bottom).

boundary motion.

Prior to solving the equations using a space–time slab Q_n, we use a least-squares projection for each prescribed node A as follows:

$$\int_{t_n}^{t_{n+1}} \boldsymbol{R}_A^h \cdot \left(\mathbf{u}_A^h - \frac{\mathrm{d}\mathbf{x}_A^h}{\mathrm{d}t} \right) \mathrm{d}t = 0, \tag{3.31}$$

where $\boldsymbol{R}_A^h$ represents the test function, $\mathbf{u}_A^h$ is represented by temporal-control velocities (unknown) and the corresponding basis functions in time, and the mesh velocity is obtained by the derivative of the mesh displacement, which is also represented by

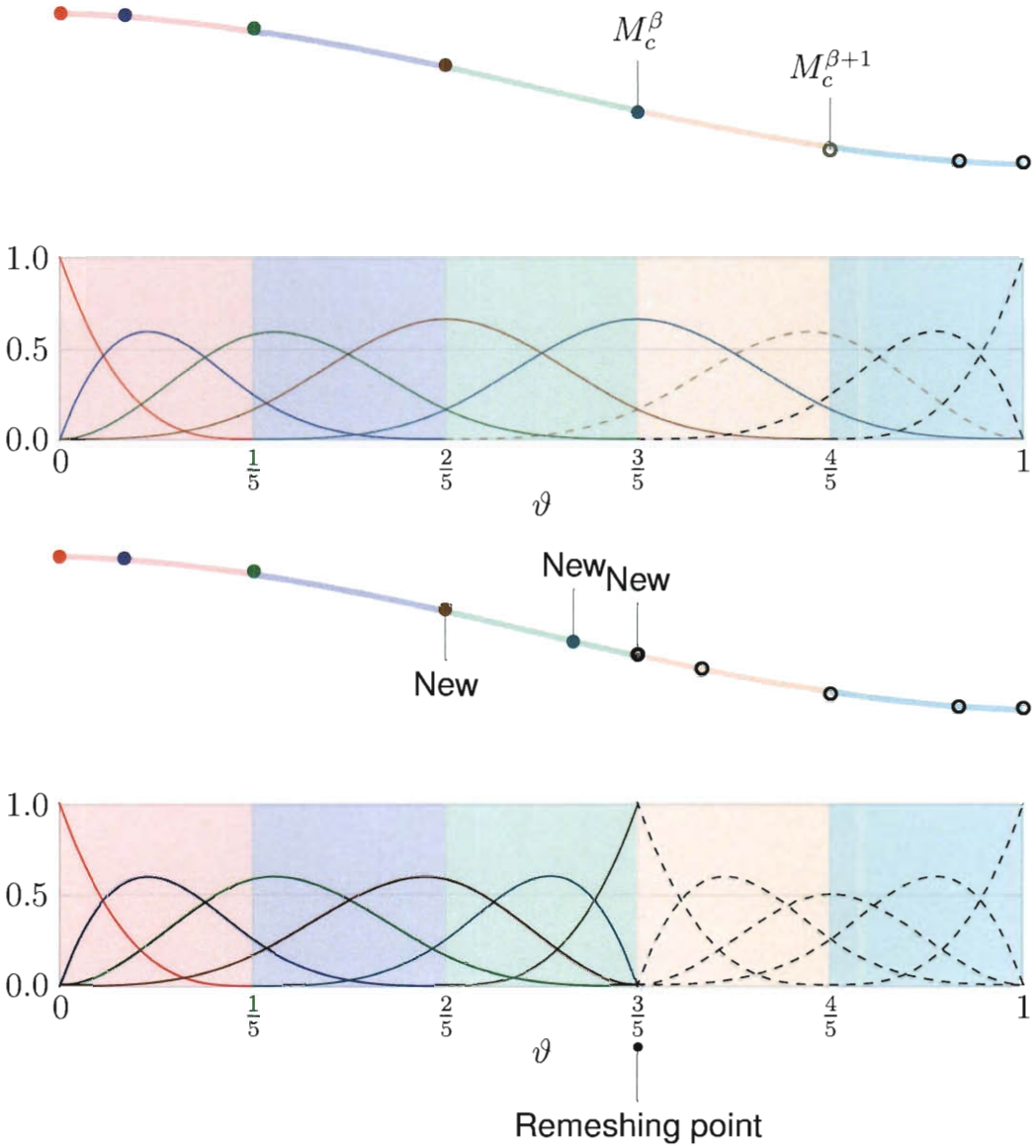


Figure 3.8: Remeshing with knot insertion. For the set of un-remeshed meshes, there are p newly-defined basis functions and the corresponding control points are marked "New". We carry out the mesh moving computations for those meshes.

temporal-control positions and their basis functions. We note that $\mathbf{u}_A^h$ at time t_n approaching from below and above might be different.

3.4.2 Starting Condition

Starting a fluid dynamics computation is not always easy, especially in the presence of moving boundaries. For example, we have developed pre-FSI computation techniques to build a starting condition for FSI computations. Here we propose a new technique as a pre-computation sequence for flow computations with prescribed boundary motion.

Suppose we want to compute with a mesh temporally represented with NURBS $(M_c^0,\ M_c^1,\ \cdots)$. We generate two additional meshes as follows:

$$M_c^{-1} = \frac{M_c^0 - \alpha M_c^1}{1-\alpha}, \tag{3.32}$$

$$M_c^{-2} = M_c^{-1}, \tag{3.33}$$

where $0 < \alpha < 1$ is an extrapolation parameter. The mesh M_c^{-1} is an extrapolation. The corresponding temporal-control point for M_c^{-1} is

$$t_c^{-1} = \frac{t_c^0 - \alpha t_c^1}{1-\alpha}, \tag{3.34}$$

with the only requirement being $t_c^{-2} < t_c^{-1}$, which determines the length of the pre-computation. For the computations reported in this paper, in temporal representation of the mesh, as NURBS basis functions, we use quadratic B-spline functions defined by the knot vector $\{0,0,0,1,1,1\}$.

Remark 12 *Based on our preliminary computations, we propose using even higher-order NURBS basis functions, so that the acceleration is continuous.*

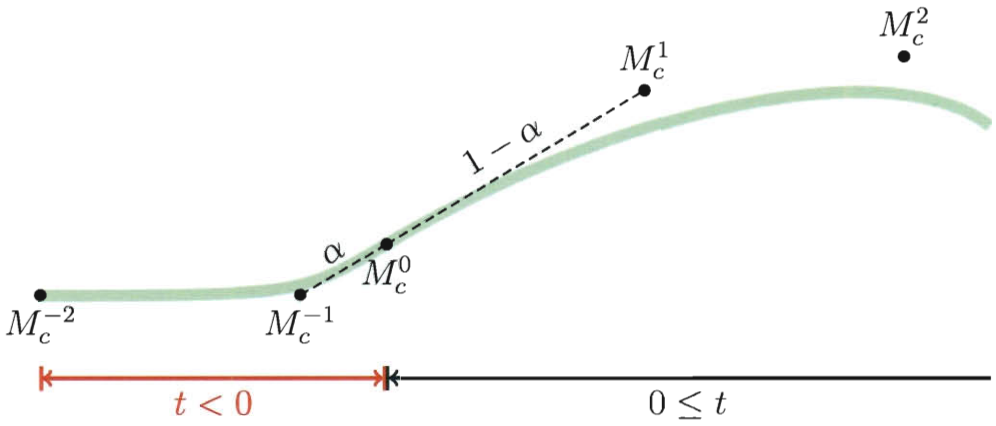


Figure 3.9: Mesh representation for the starting condition.

Chapter 4

Preliminary Computation

This chapter reports a preliminary flapping flight computation based on the method presented in Chapter 2 and the techniques presented in Chapter 3. The semi-empirically-based motion in this computation serves to represent a simplified straight-flight behavior. Mesh generation, motion representation and mesh motion are discussed in Sections 4.1, 4.2 and 4.3, respectively. Results from this computation can be found in Section 4.4. The contents of this chapter are nearly identical those reported by this author in Section 6 of [43].

4.1 Surface and Volume Meshes

Based on a digital, scanned copy of locust wings, we construct a surface mesh of the forewing (FW) and hindwing (HW) using NURBS. The FW is modeled with a single, degenerated patch. The HW is modeled with two patches — one with and one without degeneration. There are 21 and 51 control points for the FW and HW, respectively (see Figure 4.1).

We also generate a surface mesh of the locust body using 16 NURBS patches. We base the mesh on empirical height and width measurements provided at five cross-sectional positions and estimate the axial curvature of the body using video of

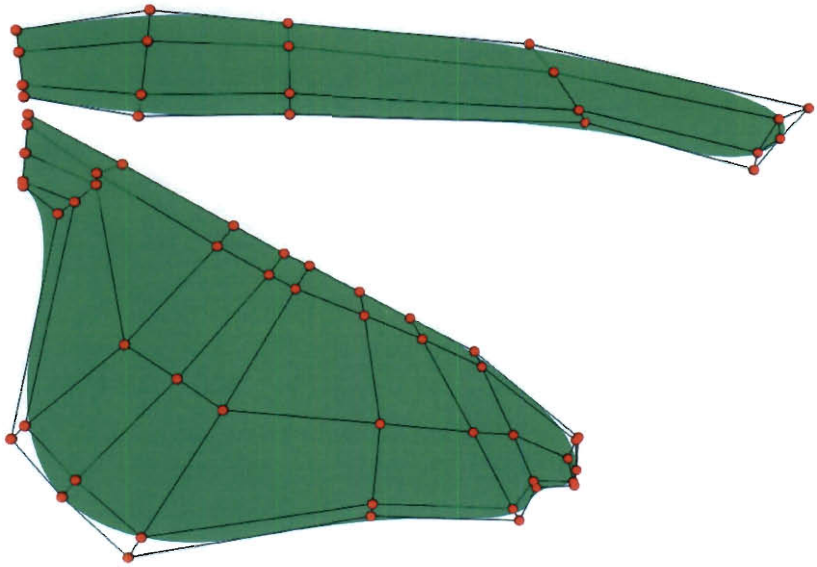


Figure 4.1: Forewing (FW) and hindwing (HW) surfaces represented by NURBS and the control points.

the locust flying. After the wing spatial-control meshes are deformed to the various positions in the flapping motion as we will describe in Section 4.2, we discretize them at each temporal-control point. The triangular surface mesh used in the computation is shown in Figure 4.2.

Figure 4.2: Wing and body surface meshes with triangular elements.

For automatic volume mesh generation, the wings have a finite thickness of 1% of the FW root chord, which tapers to zero thickness at the wing edges. We generate a one-layer refinement region near the wing surface. In this region, the element height is 10% of the FW root chord. In addition, we have a cylindrical region of increased refinement around the locust. We also specify increased volume mesh refinement in the region between the FW and HW. The volume mesh within this cylindrical region is then generated with tetrahedral elements using an automatic mesh generator. Next we define a box that contains the cylindrical region, and we generate a tetrahedral mesh in that; again using an automatic mesh generator. We rotate this mesh to an angle representing the approximate body angle of the locust within the full computational domain. A volume mesh within the full domain is then also automatically generated with tetrahedral elements. The number of nodes and elements in the volume mesh varies between each temporal patch. The average number of nodes and elements in these meshes are approximately 430,000 and 2.6-million, respectively. The volume mesh and refinement regions are shown in Figure 4.3.

4.2 Flapping-Motion Representation

We must also provide a prescribed motion that is representative of straight-flight flapping. Here we face the challenge of reconciling data acquired through photogrammetry with data that is suitable as an input for computational analysis. We use 4, 1 and 10 wind-tunnel-acquired tracking points for the body, HW (wingtip) and FW (1 common with the body). This motion is reflected across the sagittal plane of the locust to create symmetric flapping motion. We generate additional HW "tracking points" to deform the wing in a similar manner to that observed in wind tunnel videos. In total, we use 76 tracking points to represent the desired wing motion.

Next, we temporally interpolate our representative data set. For each tracking point, we apply a temporal NURBS representation, as discussed in Sections 3.1.3

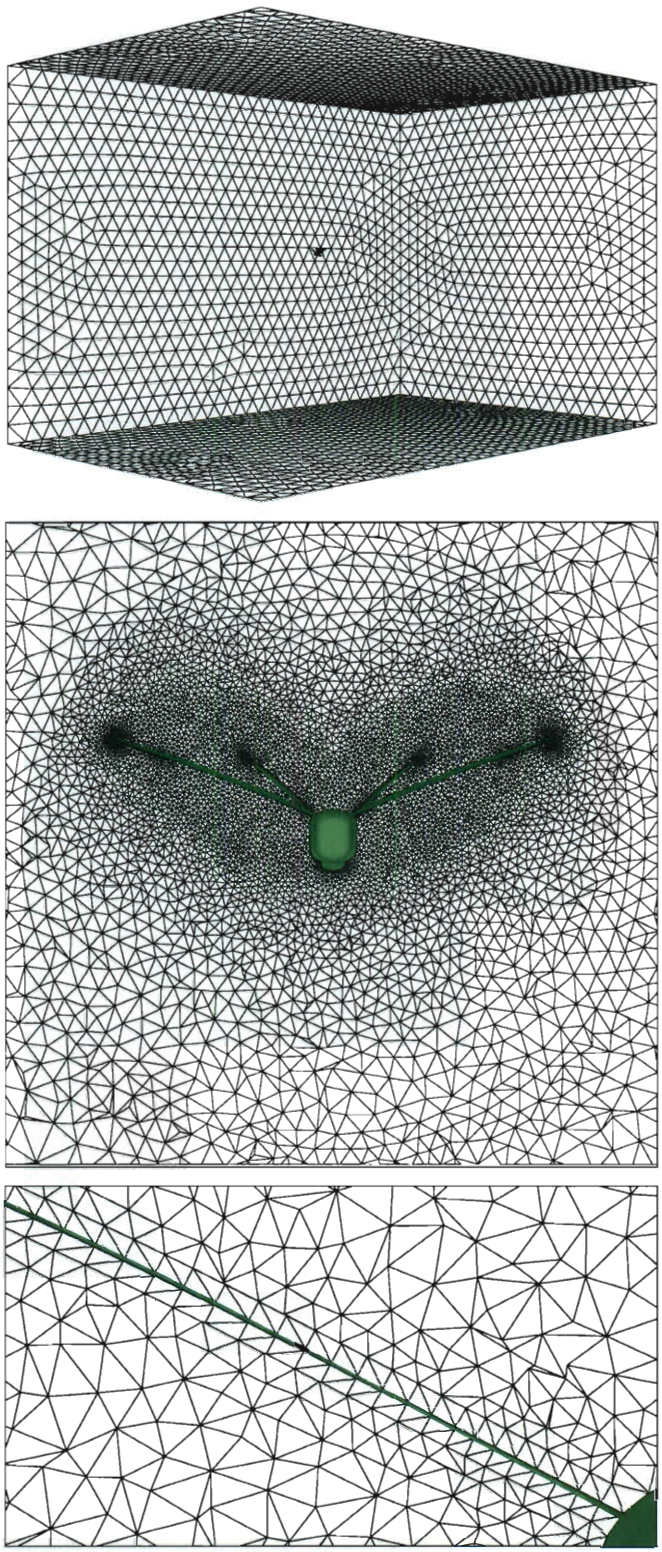


Figure 4.3: Volume mesh shown for the full computational domain (top), cylindral-refinement region (middle), and refinement region near the wing surface (bottom).

and 3.1.4. We use quadratic B-splines with temporal-control points as defined by Eq. (3.8), and we reduce 171 sampling points to 60 temporal-control points by using Eq. (3.9).

Then, we spatially represent the tracking points at each temporal-control point. Spatial interpolation is accomplished using the SSDM described in Section 3.2. The FW and HW SS consist of 6 and 9 control points, respectively. An illustration of this process for the left HW is shown in Figure 4.4.

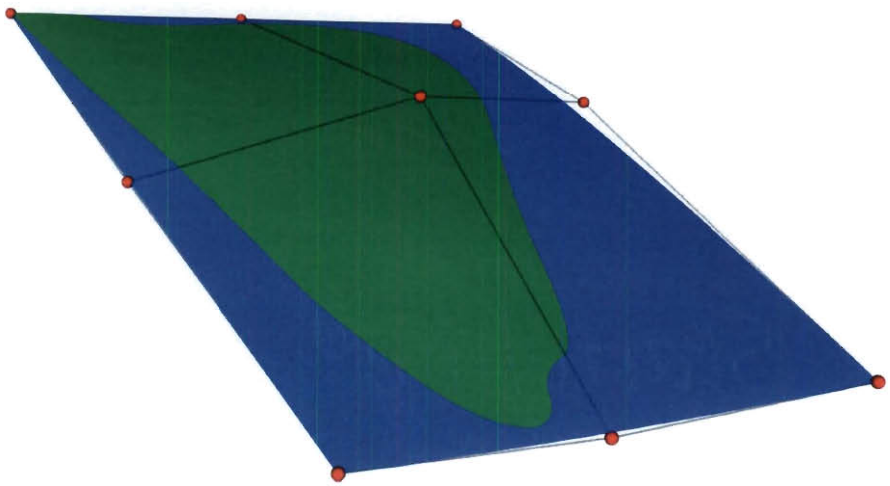


Figure 4.4: Deformed SS and associated control points along with the projected HW NURBS surface.

In the least-squares fit from tracking points to SS control points, the control points nearest to the body are fixed. To minimize the effect of an unrealistic least-squares fit due to the single tracking point at the tip, additional points are generated using linear extrapolation between the outermost tracking points. These additional points are included in the 76 total points mentioned earlier. At each temporal-control point, a final least-squares projection is then performed between each SS and corresponding wing surface defined by the NURBS mesh. Now, we have a NURBS-represented data set in both space and time for each wing as illustrated for the FW in Figure 4.5.

The motion of the wings requires that we remesh at some time during the flapping

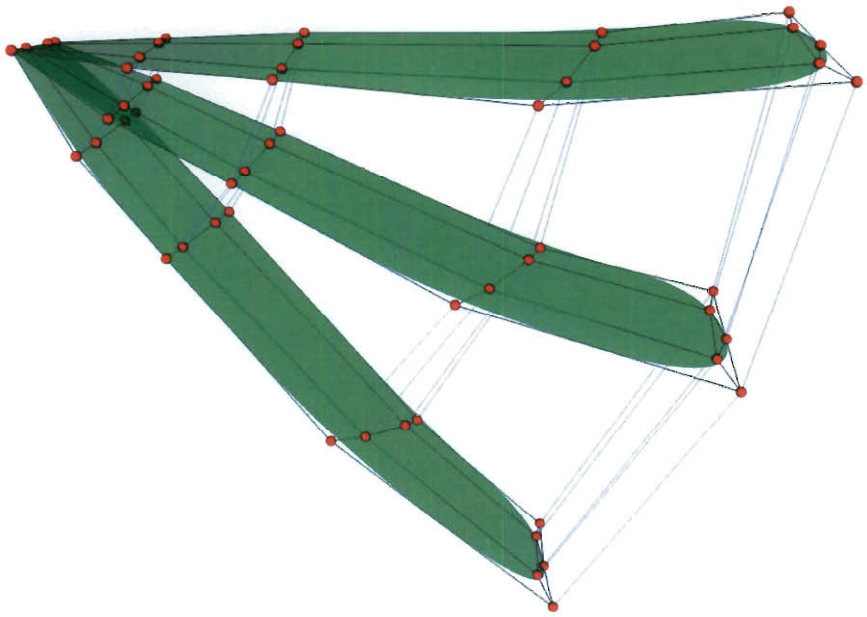


Figure 4.5: FW control mesh and corresponding surface at three temporal-control points.

cycle. To facilitate remeshing, we use temporal knot insertion (see Section 3.3.2) to create multiple equally-spaced temporal patches prior to volume meshing. Each temporal patch contains 5 control points. We note that the spatial position corresponding to the last control point of each temporal patch is identical to that of the first control point in the next patch. Within each temporal patch, we select the middle control point and generate a volume mesh.

4.3 Mesh Motion

To capture the wing motion and deformation within each temporal patch, the volume mesh inside the box must be deformed to the corresponding temporal-control surface mesh. Due to the relatively large change in deformation between each temporal-control point, we use subiterations for the mesh computation to divide the steps between temporal-control points into 20 smaller steps. We move the mesh, which

corresponds to the middle control point, backward and forward through each smaller step using 1,500 GMRES iterations. Using this approach, as shown in Figure 4.6, the worst mesh quality occurs at the beginning and end of each temporal patch.

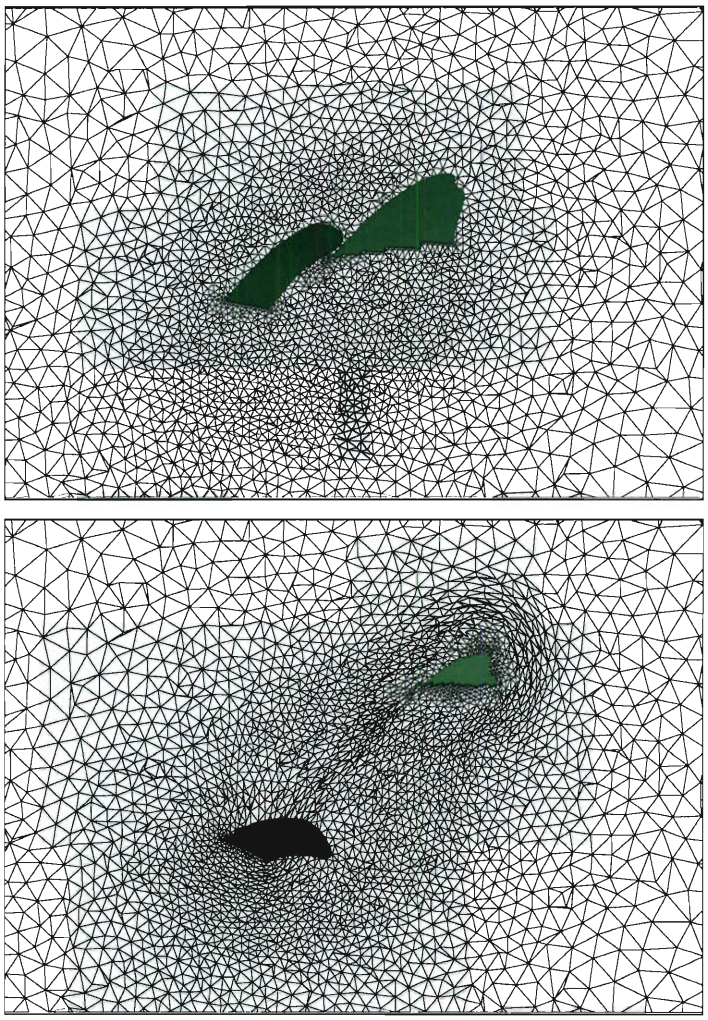


Figure 4.6: The volume mesh obtained by the automatic mesh generator (top) and after being moved to the first temporal-control point of that patch (bottom).

We use a fluid dynamics starting condition as described in Section 3.4.2. We obtain the temporal-control meshes M_c^{-1} and M_c^{-2} by using Eqs. (3.32) and (3.33). This starting condition can be seen in the gray region of Figure 4.7.

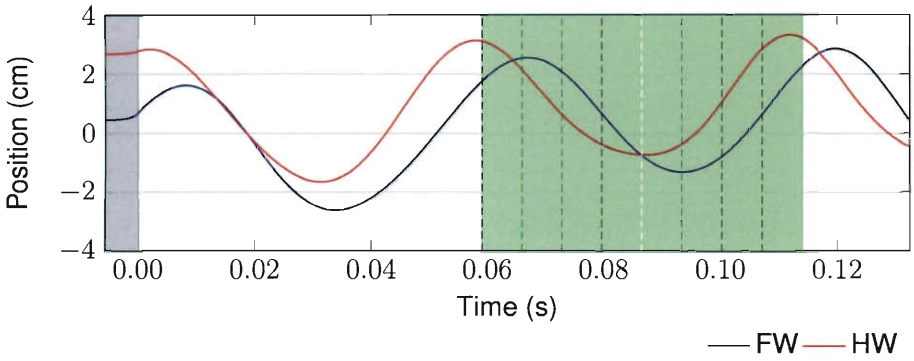


Figure 4.7: FW and HW tip position in time with the shaded regions showing the extrapolation region (gray) and section used for results visualization (green). Dashed, vertical lines indicate the points in the cycle used in Figures 4.8, 4.9, and 4.10.

4.4 Fluid Computation

Prior to beginning the prescribed flapping motion, we compute for 200 time steps to develop the flowfield with the mesh M_c^{-2}. Over the first 100 time steps of this computation, we use a Cosine form to smoothly increase the inflow velocity from 0 to 2.4 m/s, which represents the average wind tunnel velocity. We then begin prescribing the motion using the method described in Section 3.4.2.

For this computation, we use 25 space–time slabs (with linear basis functions) for each of the 3 knot spans in the temporal representation of the mesh, which results in a remeshing frequency of every 75 time steps. This results in a time-step size of 2.2×10^{-4} s. We use 4 nonlinear iterations per time step. The DSD/SST-SUPS and DSD/SST-VMST techniques (as described in [49]) are used for the first two and last two nonlinear iterations, resepectively. The stabilization parameters are those given by Eqs. (7)–(11) in [62] for $\tau_{\text{M}} = \tau_{\text{SUPG}}$, and ν_{C} is defined as follows:

$$\nu_{\text{C}} = \left(\nu_{\text{LSIC}}^{-2} + \nu_{\text{HRGN}}^{-2}\right)^{-\frac{1}{2}}, \tag{4.1}$$

$$\nu_{\text{LSIC}} = \tau_{\text{SUPG}} \left\|\boldsymbol{u}^h - \mathbf{v}^h\right\|^2, \tag{4.2}$$

$$\nu_{\text{HRGN}} = \frac{h_{\text{RGN}}^2}{\tau_{\text{SUPG}}}, \tag{4.3}$$

where h_{RGN} is given by Eqs. (10) and (11) in [62]. We compute the stabilization parameters after the predictor step and after the first two nonlinear iterations. The number of GMRES iterations for the nonlinear iterations are 30, 60, 60 and 120.

Remark 13 *Eq. (4.3), which comes from [6], has been modified for compatiblity with other stabilization parameters.*

Remark 14 *We have seldom observed close-to-zero or negative diagonal terms when the time-step size is large. This occurs when the fine-scale velocity is much larger than the coarse-scale (discrete) velocity. We believe this is due to the predictor, which assumes all velocities are the same from the previous time step except those with Dirichlet conditions.*

Because the wing velocity varies during the stroke, the Reynolds number range is 1,000–2,500 as calculated at 75% of the wing span (measured from root chord to tip). Figures 4.8–4.10 show the results form the preliminary computations.

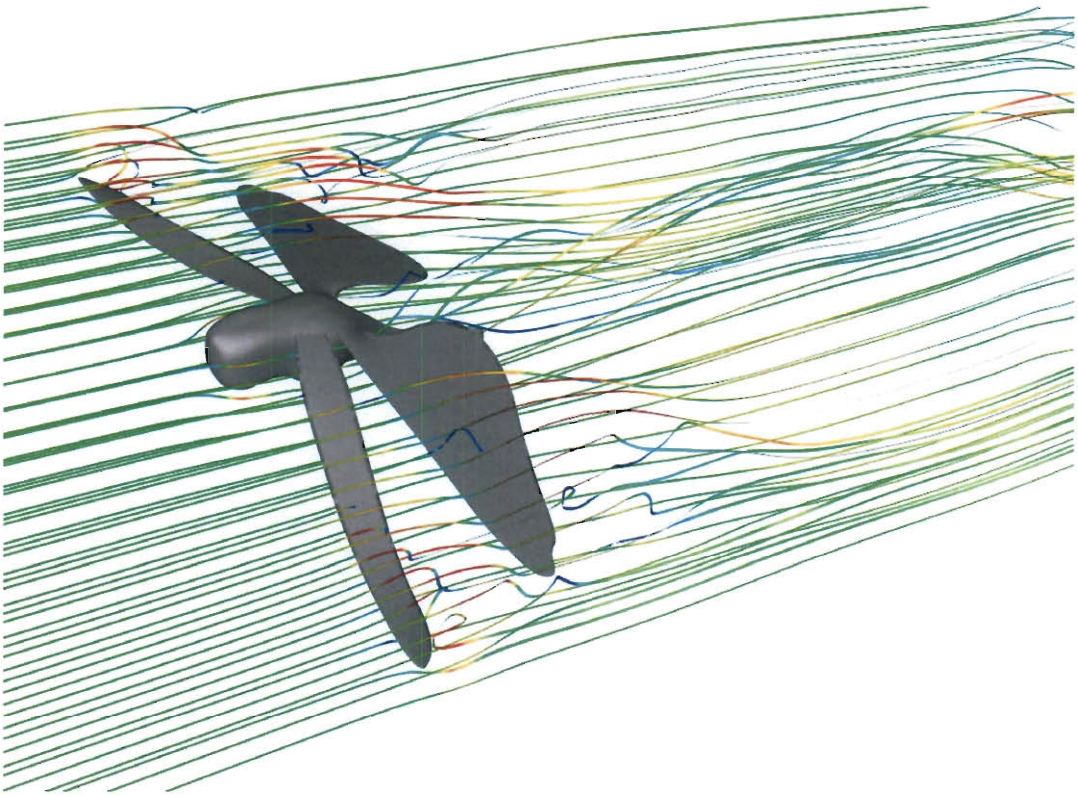


Figure 4.8: Streamlines colored by velocity at the time indicated by the vertical, white dashed line in Figure 4.7.

Figure 4.9: Voriticty at eight points during the flapping cycle (left to right, top to bottom) indicated by the vertical lines in Figure 4.7.

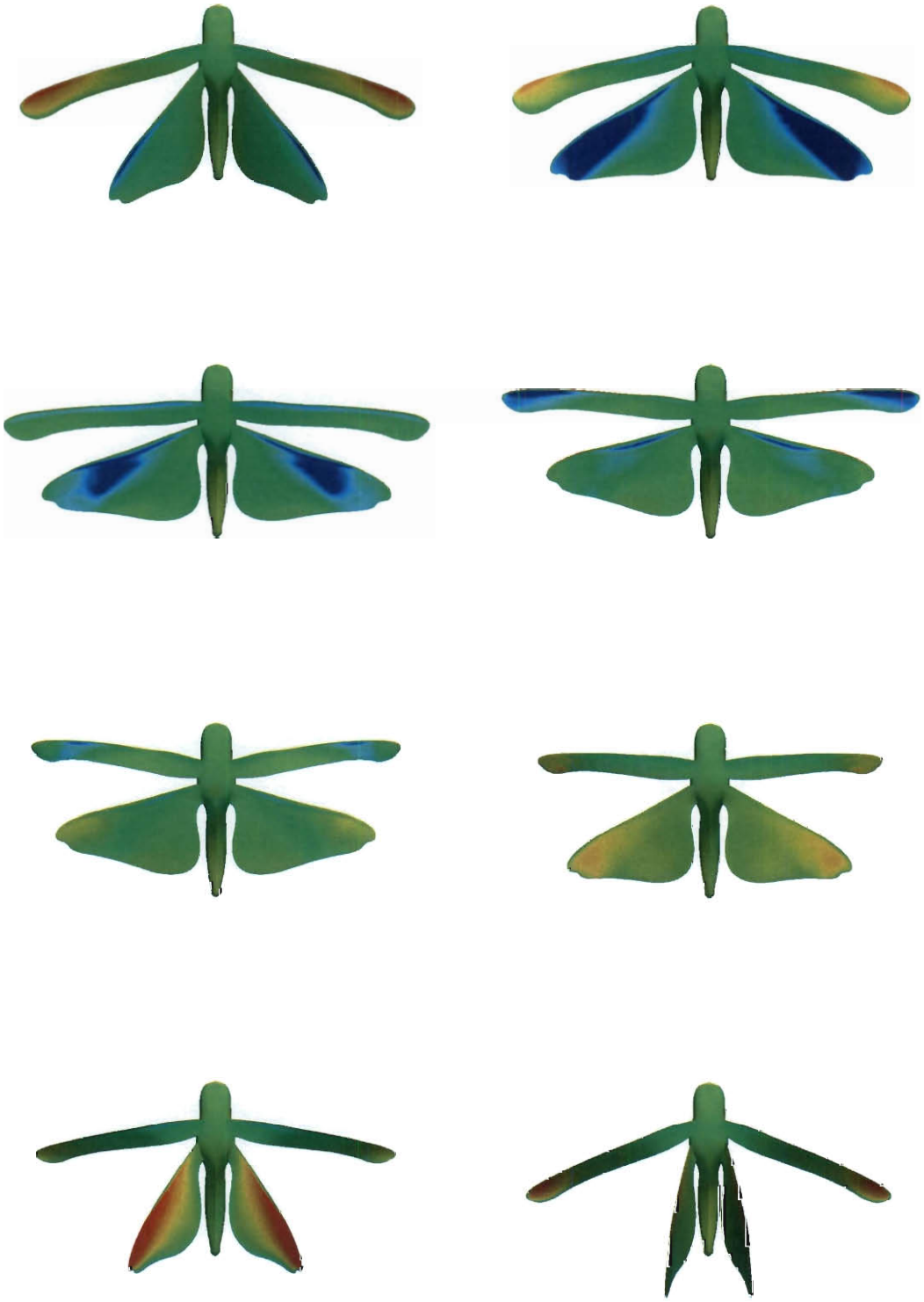


Figure 4.10: Surface pressures at eight points during the flapping cycle (left to right, top to bottom) indicated by the vertical lines in Figure 4.7.

Chapter 5

Computation with Improved Temporal Representations

The preliminary computation presented in Chapter 4 served as a step toward achieving an accurate fluid simulation of flapping flight aerodynamics, and a number of lessons were learned in the process of completing this computation. Building upon these lessons, Chapter 5 presents the next step toward this goal. While similar to the simulation presented in Chapter 4, this computation includes higher-order basis functions in the temporal interpolation as well as a more realistic and periodic flapping motion.

5.1 Setup

As previously noted, the setup for the computation presented in this chapter is very similar to that of the computation presented in Chapter 4. The major differences, which are explained in detail in Sections 5.1.1, 5.1.2, and 5.1.3, occur in the wing flapping motion representation. Additionally, a summary of the computational setup is provided in Section 5.1.4.

5.1.1 Higher-Order Interpolation

As stated in Remark 12 in Section 3.4.2, higher-order basis functions should be used in temporal interpolation to achieve a continuous acceleration. In this computation, we use third-order, or cubic, basis functions for temporal interpolation. Based on information presented in Section 1.3, cubic basis functions will maintain C^2–continuity across knots. Thus, acceleration will be continuous across temporal knots.

The process of moving from quadratic to cubic basis functions for temporal interpolation is fairly straightforward. We design the temporal B-spline basis functions to satisfy Eq. (3.8). As in the case of quadratic basis functions, we utilize a least-squares projection to translate from a linear finite element mesh consisting of two-node elements to a B-spline representation.

5.1.2 New Motion

When the flapping motion of the Chapter 4 computation was compared with video of a locust flying in a wind tunnel, we observed that some of the details of the flapping motion in our model, especially in the HW, were significantly different from the video. A variety of improvements were made to the flapping motion used in this chapter to make it better represent locust flight.

For this computation, we use a more recent data set provided by our collaborators at BCM. This new data set includes 2 more tracking points for each HW than the data used in the previous computation. The tracking points provided in this data set can be seen in Figure 5.1 With the help of these tracking points, we generate additional "tracking points" to more closely represent locust flight characteristics observed in the video. As done in Chapter 4, one side is selected and reflected to create a symmetric flapping model.

We also automate our technique for mapping to the SS by using the Newton–Raphson method. For mapping the tracking points to the SS, this involves finding

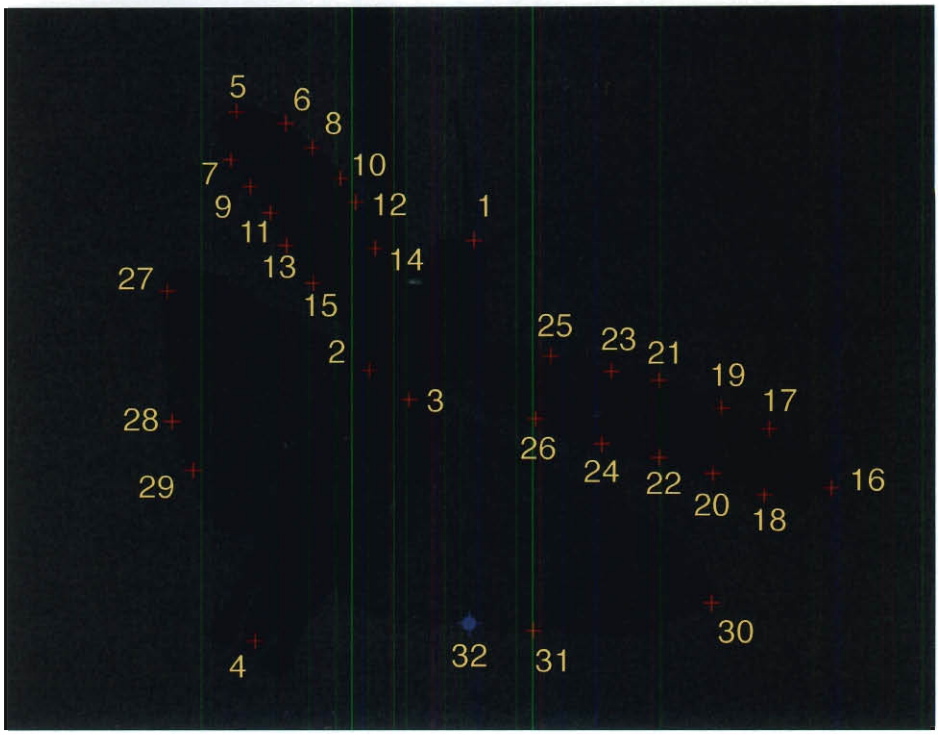


Figure 5.1: Tracking points in data set provided by BCM used in Chapter 5 computation. Image provided by BCM collaborators.

the location in parametric space that corresponds to a given location in physical space; for mapping between the wing surface and the SS, we use the physical location of the wing quadrature points. The mapping between the tracking points and the SS and the mapping between the SS and the wing surface for the FW and HW can be found in Figure 5.2. Unlike the technique used in Chapter 4, we specify the control points of the SS near the wingtip as prescribed conditions for the least-squares fit of the tracking points and the SS. This is in addition to prescribing the control points of the SS near the root chord. This technique provides a better representation of the SS motion.

5.1.3 Periodic Motion

While flapping motion kinematics are inherently cyclical, they are not necessarily periodic. This can be seen in the wingtip trace shown in Figure 4.7 in that for the

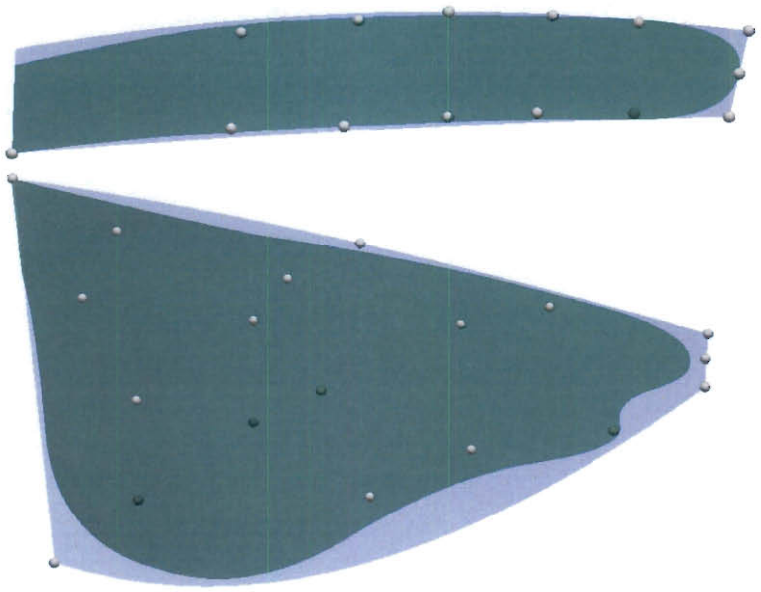


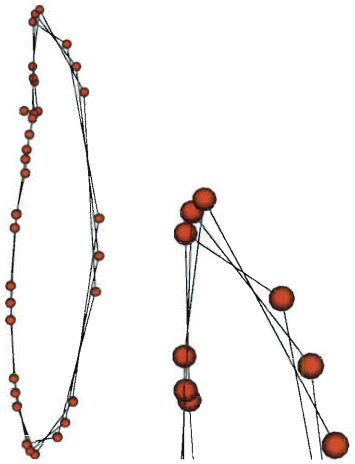
Figure 5.2: Tracking points, SS, and wing surface from the new mapping used in this computation shown at one temporal control point.

FW, the vertical position at the start of the first downstroke does not correspond to exactly the same position for subsequent strokes. This, of course, adds to the difficulty of computing flapping flight aerodynamics. It is beneficial to have a periodic data set for a single flap cycle, where the first and last points of the cycle are co-located. Thus, a single set of deformed meshes can be appended to produce as many flapping cycles as are required. Additionally, rather than using the procedure discussed in Section 3.4.2 for obtaining a starting condition, because of the periodicity of the motion we can build a starting condition by performing starting–point computations over a desired number of cycles.

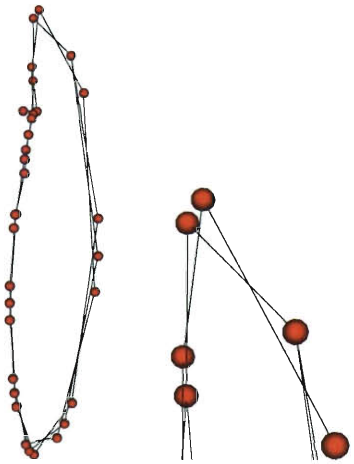
To achieve the most realistic periodic data set, the following procedure was used. From the data set provided by our BCM collaborators, which was discussed in Section 5.1.2, we extract a single flapping cycle and match two points at the end of the cycle by averaging. This cycle is repeated twice to form a motion of three identical cycles and then used as the input for the least-squares projection described in

Section 5.1.1.

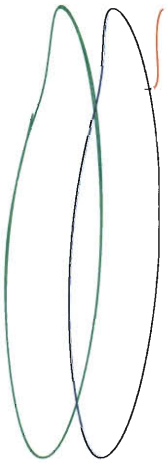
We now extract one cycle by inserting knots at the top of the hindwing cycle (end of the upstroke and beginning of the downstroke). To maintain continuity, the three control points, which correspond to the knot at which we intend to insert additional knots should be co-located. Thus, we average the positions of these three control points for the cycle of interest. Finally, we insert knots to extract a single cycle. These steps can be seen in Figure 5.3.



(a) Original control points for 3 identical flapping cycles (enlarged view of top of cycle)



(b) Control points after averaging at top of cycle (enlarged view showing averaged control points)



(c) The three sections of path defined by (a) after knot insertion (expanded for emphasis)

(d) A single, periodic cycle after extraction with control points

Figure 5.3: Process used to make a single periodic cycle.

5.1.4 Summary of Setup

Like in Chapter 4, knot insertion is used to divide the flapping cycle into temporal patches and meshes are generated as before. We note one difference from the mesh generation in the previous computation: in this computation, body angle, the angle between the locust's body and the freestream velocity, is accounted for by rotating the cylindrical refinement region instead of the "box mesh" (see Section 4.1) as done before. This eliminates the need for the box volume mesh used previously. The cylindrical refinement region and domain boundaries are shown in Figure 5.4. We

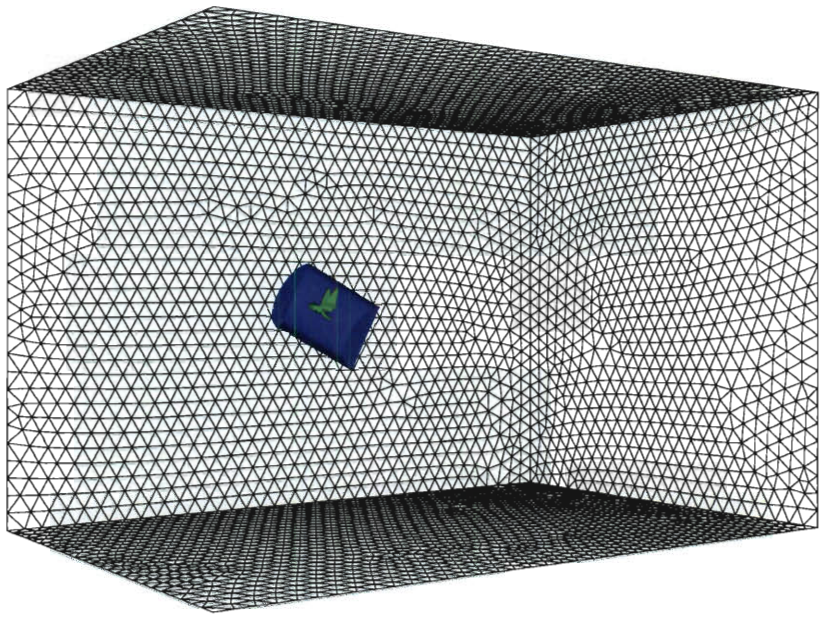


Figure 5.4: Computational domain boundaries and cylindrical refinement region, which has been rotated to account for in-flight body angle.

note that the computational domain used in this computation is slightly smaller than the one used in Chapter 4. The additional elements in the original domain are not essential to the solution and add to cost of the computation.

We use knot insertion to divide the cycle into 4 temporal patches for remeshing. The path of the HW wingtip through each of these temporal patches and the relevant parameters within each patch are shown in Figure 5.5 and Table 5.1, respectively. As shown in Figure 5.5, we select the temporal patch boundaries in such a way that the

distance traveled by the HW in each patch is approximately equal. This, however, results in an unequal number of temporal control points in each temporal patch as shown in Table 5.1.

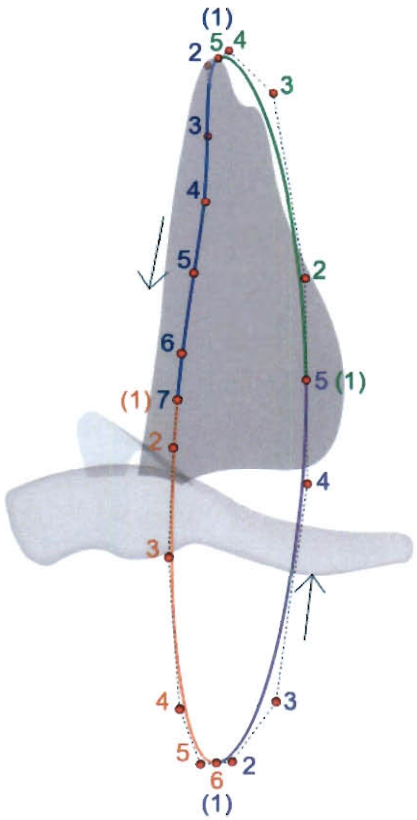


Figure 5.5: HW wingtip trajectory with temporal patches and control point numbering (local to each patch). Co-located control points, which exist at the end of one patch and start of the next, are indicated by parentheses.

Temporal Patch	Figure 5.5 Color	Control Points	Knot Spans	Meshing Point	Number of Nodes	Number of Elements
1	Blue	7	4	4	355,229	2,115,916
2	Orange	6	3	3	389,981	2,323,144
3	Purple	5	2	2	346,993	2,066,797
4	Green	5	2	3	380,034	2,264,324

Table 5.1: Summary of computational setup. In each temporal control patch (differentiated by color in Figure 5.5), we indicate the number of temporal control points and at which point we generate the tetrahedral volume mesh with the number of nodes and elements shown.

With only the few exceptions, which we note here, mesh motion and the fluid computation were accomplished with the same parameters as those described in Sections 4.3 and 4.4. In this simulation, we compute mesh motion for only the cylindrical refinement region and then merge with the rest of the volume mesh. This is in contrast to what was done in Chapter 4, where mesh motion was computed for the cylindrical and box volume mesh regions after they had been merged. This approach is more computationally efficient and is supported by the preliminary computation mesh motion, where mesh deformation appeared to be mostly limited to the cylindrical refinement region.

Prior to the beginning of the prescribed flapping motion, as done in Chapter 4, we compute 200 time steps to develop the flowfield. We use a time-step size of 1.7×10^{-4} s and 4 nonlinear iterations per time step. The DSD/SST-SUPS and DSD/SST-VMST techniques are used for the first two and last two nonlinear iterations, with the stabilization parameter as given in Chapter 4 for the prescribed-motion part. The number of GMRES iterations for the nonlinear iterations are 30, 60, 60 and 60 for the first 115 time steps and 30, 60, 60 and 120 for the last 85. As previously mentioned, the temporal periodicity of this computation allows us to use the output of one cycle as an initial condition for the next. Thus, we compute for one complete cycle to establish a starting condition for the results reported in Section 5.2. In this first cycle, we use the same parameters as those used in the second stage of developing the flowfield.

Although not discussed in Chapter 4, we partition this mesh in the same way as done in the previous computation to enhance parallel efficiency. Mesh partitioning is based on the METIS [30] algorithm. We partition for 128 processors and compute on Rice's BlueBioU - IBM POWER 7 Bioscience Computing Core using either 128 or 256 processors.

5.2 Results

We compute two periodic flapping cycles and present the results of that computation in this section. As done in Section 4.4, for this computation, we use 4 nonlinear iterations per time step and a time step size of 1.7×10^{-4} s. The DSD/SST-SUPS and DSS/SST-VMST techniques are used for the first two and last two nonlinear iterations, respectively. We also compute stabilization parameters as discussed in Section 4.4. However, the number of GMRES iterations for the nonlinear iterations are 30, 60, 120, and 180 GMRES iterations.

As previously noted, prescribing an accurate wing motion and deformation is important to achieving an accurate fluid simulation. Thus, we begin by comparing the motion of our model to that of photographs recorded from the wind tunnel upon which our model's motion is based. We compare our model to these photos at eight approximately equally-spaced points during the flapping cycle in Figures 5.6 and 5.7. We note that some differences should exist between our model and the photos due to the process used to make the flapping motion symmetric and periodic.

The flapping period in this computation is 0.047 s, which corresponds to a flapping frequency of 21.4 Hz. We show the wingtip position and lift and drag forces (pressure component only) in Figures 5.8 and 5.9. Although we compute for two complete flap cycles, we show visualizations of only the last flap cycle in Figures 5.10-5.14.

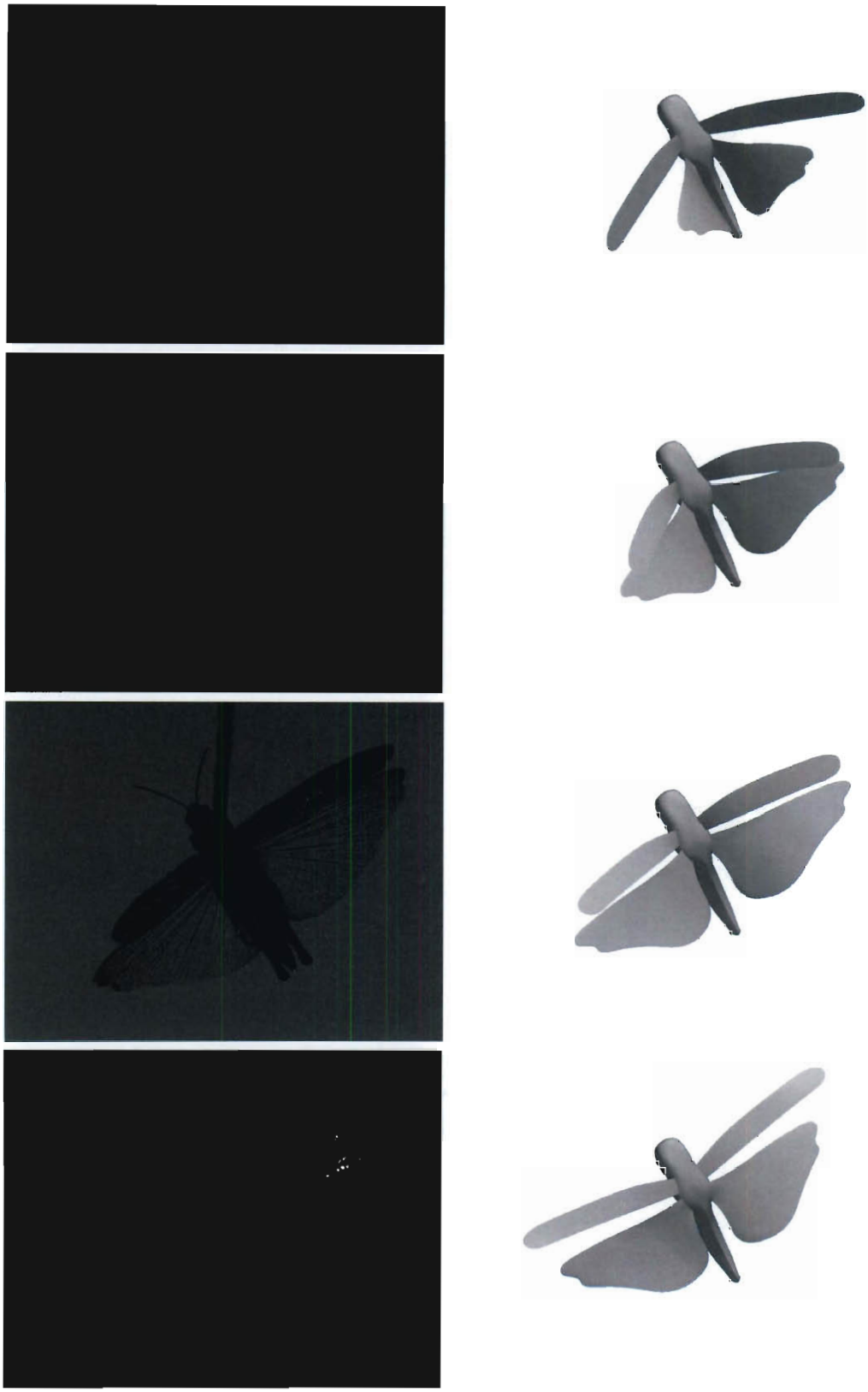


Figure 5.6: Comparison of computational model and wind tunnel photographs at first four points in time. Viewing angles are matched approximately. Wind tunnel photographs provided by BCM collaborators.

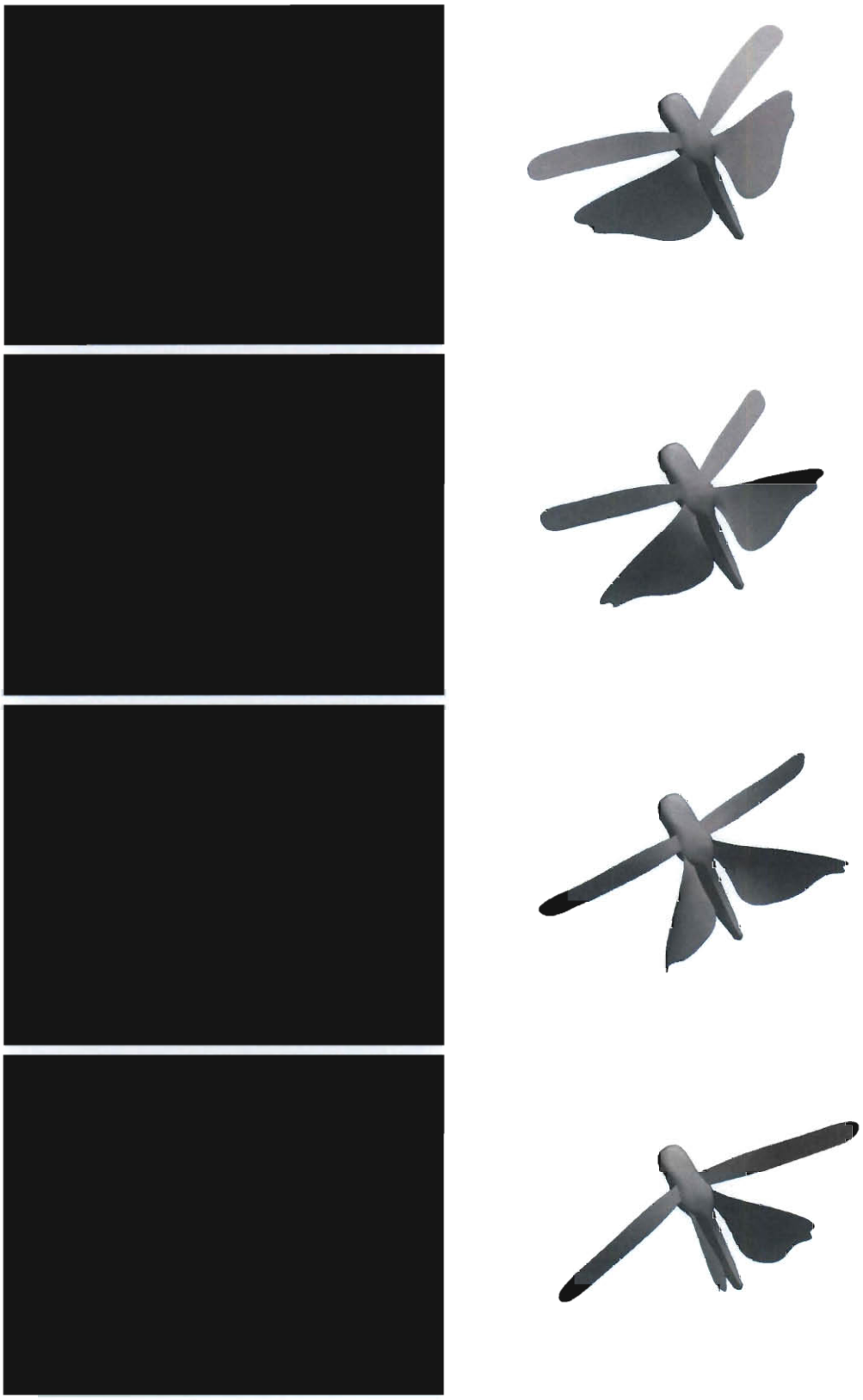


Figure 5.7: Comparison of computational model and wind tunnel photographs at last four points in time. Viewing angles are matched approximately. Wind tunnel photographs provided by BCM collaborators.

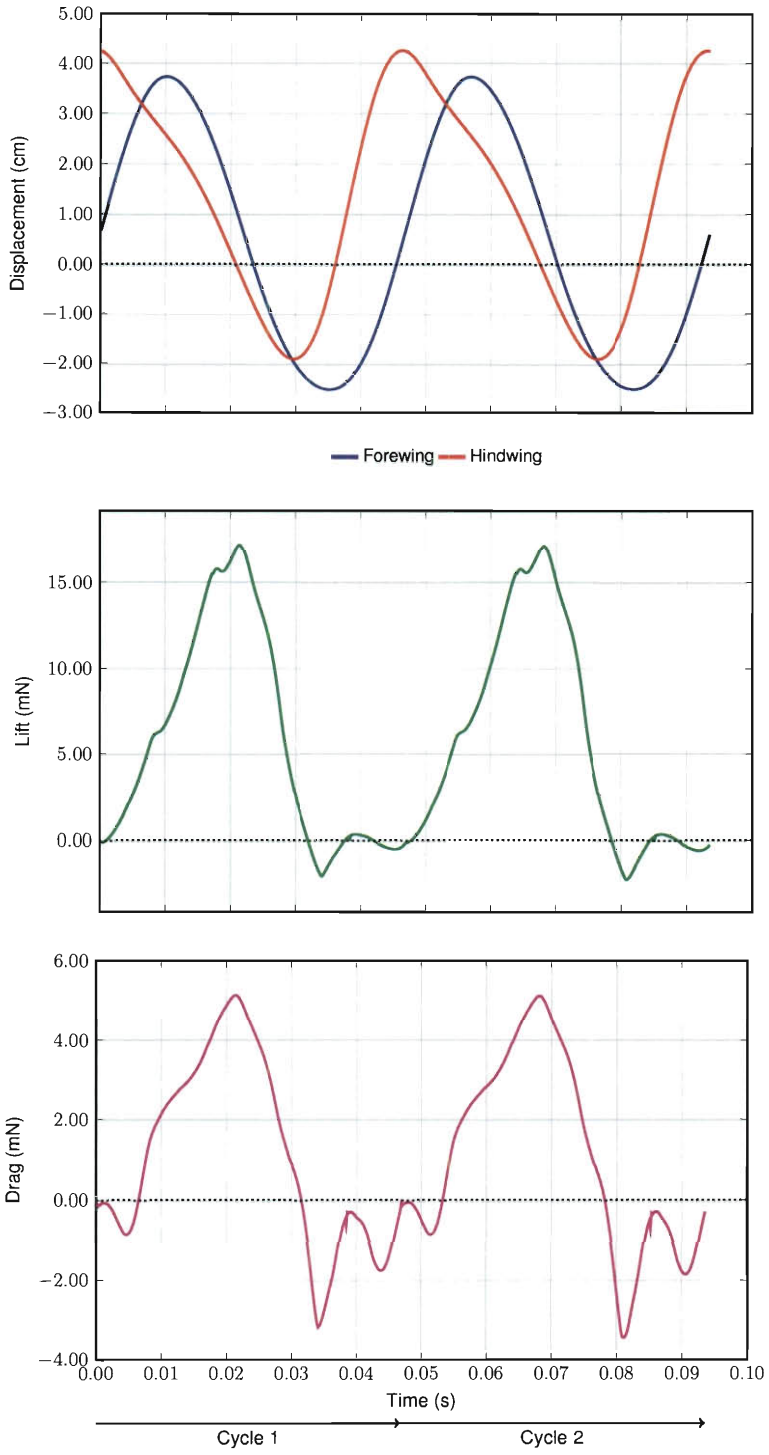


Figure 5.8: Forewing and hindwing wingtip displacement from the root chord (top), total locust lift force generated over two cycles (middle), total locust drag force, where a negative value indicates that thrust exceeds drag at that time (bottom). Note that the scales are different in the last two plots.

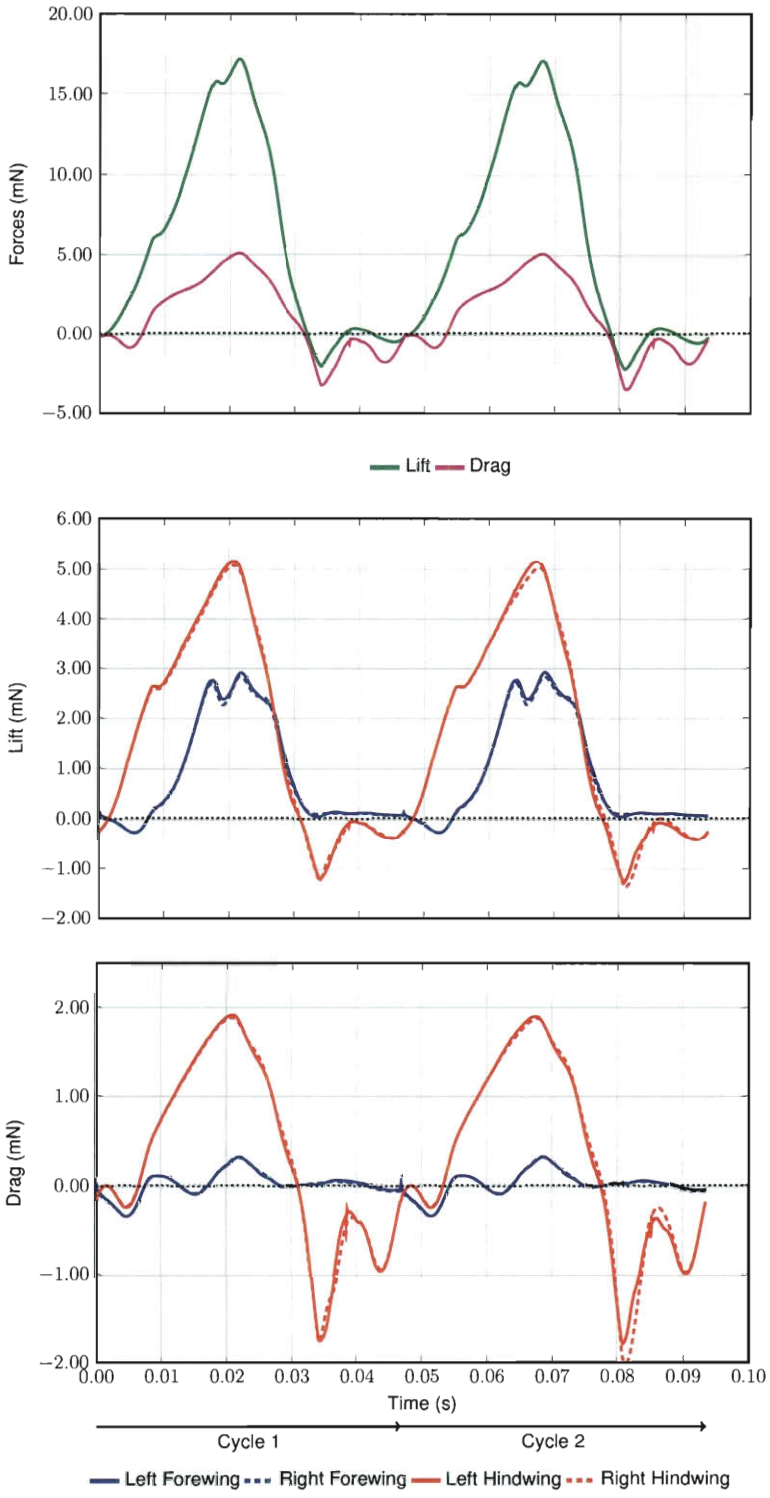


Figure 5.9: Total locust lift and drag forces shown in Figure 5.8 but plotted on the same scale (top), lift force contribution of each wing (middle), drag force contribution of each wing (bottom).

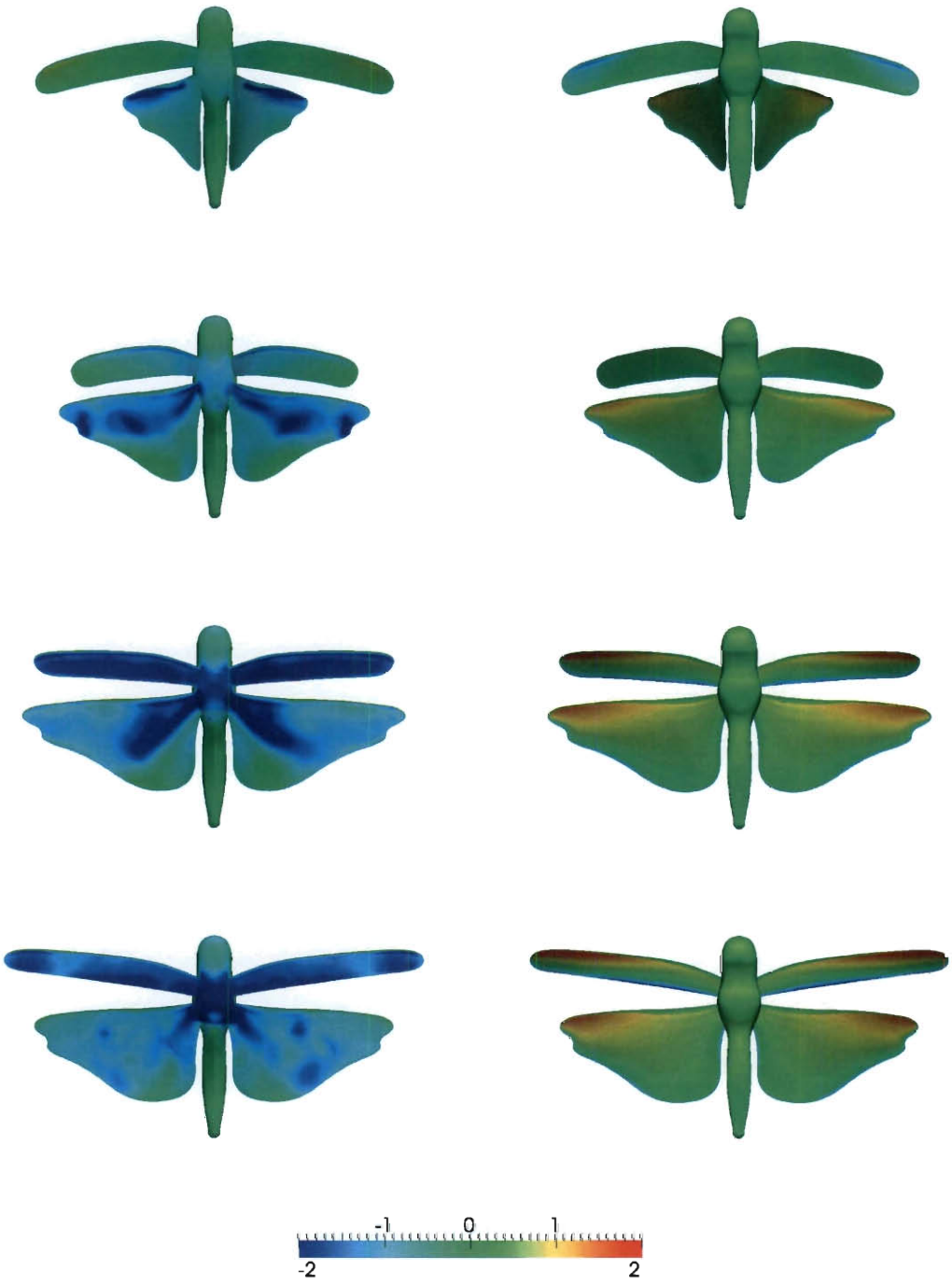


Figure 5.10: Surface pressure difference from freestream in kPa at the first four equally-spaced points during the second flapping cycle (top view on left, bottom view on right).

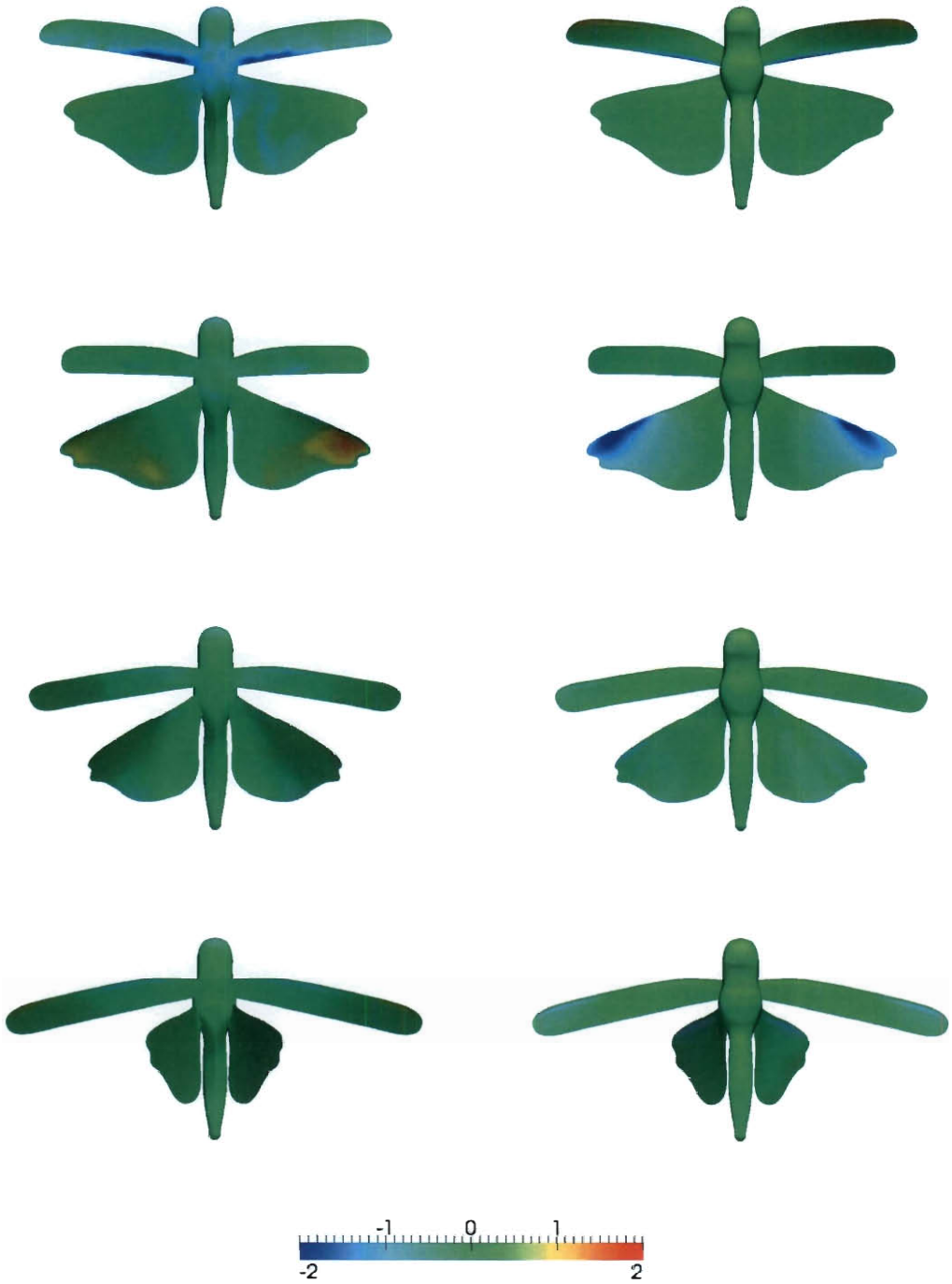


Figure 5.11: Surface pressure difference from freestream in kPa at the last four equally-spaced points during the second flapping cycle (top view on left, bottom view on right).

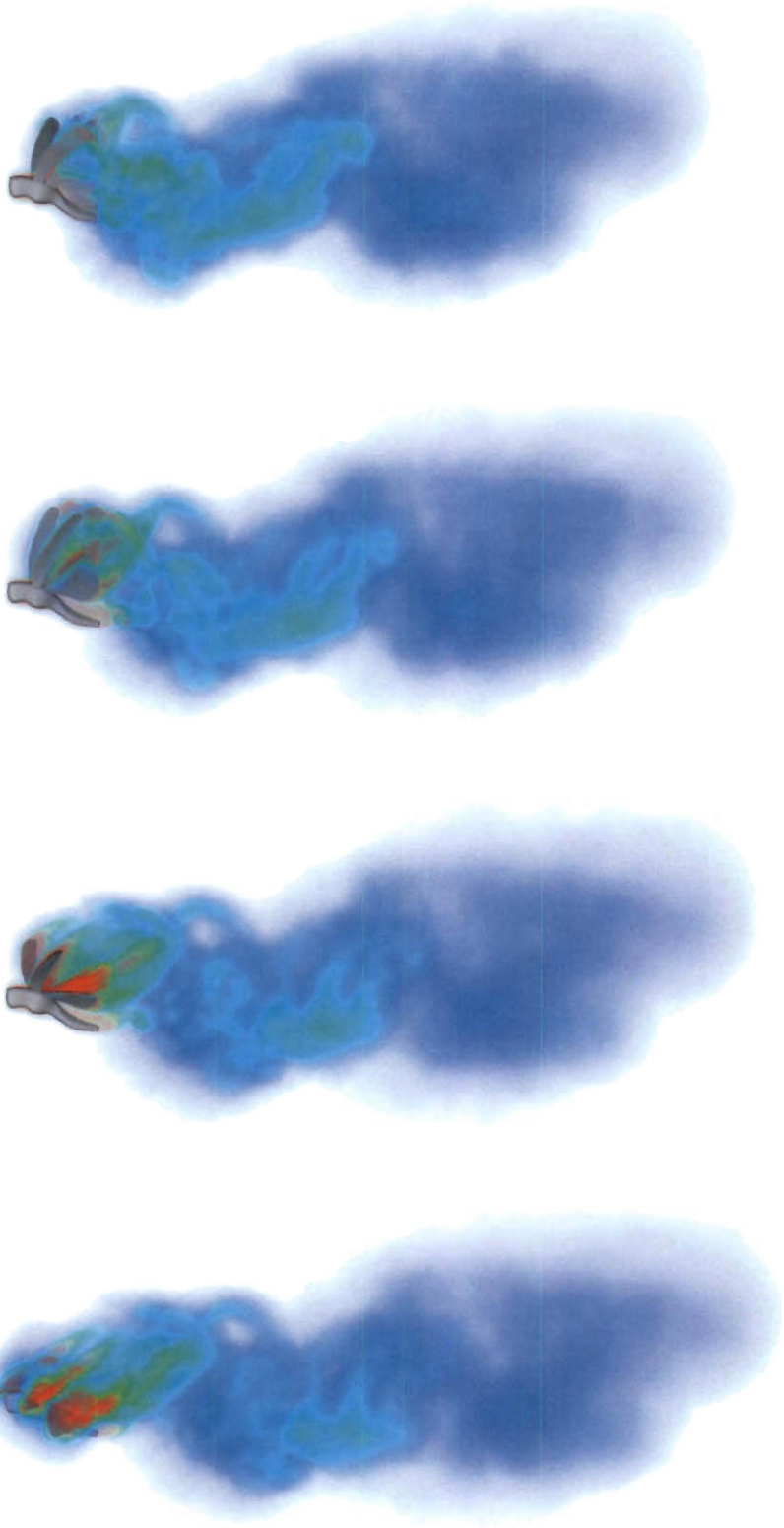


Figure 5.12: Volume rendering of vorticity for the first four approximately equally-spaced points during the second flapping cycle (top to bottom). Red indicates higher vorticity.

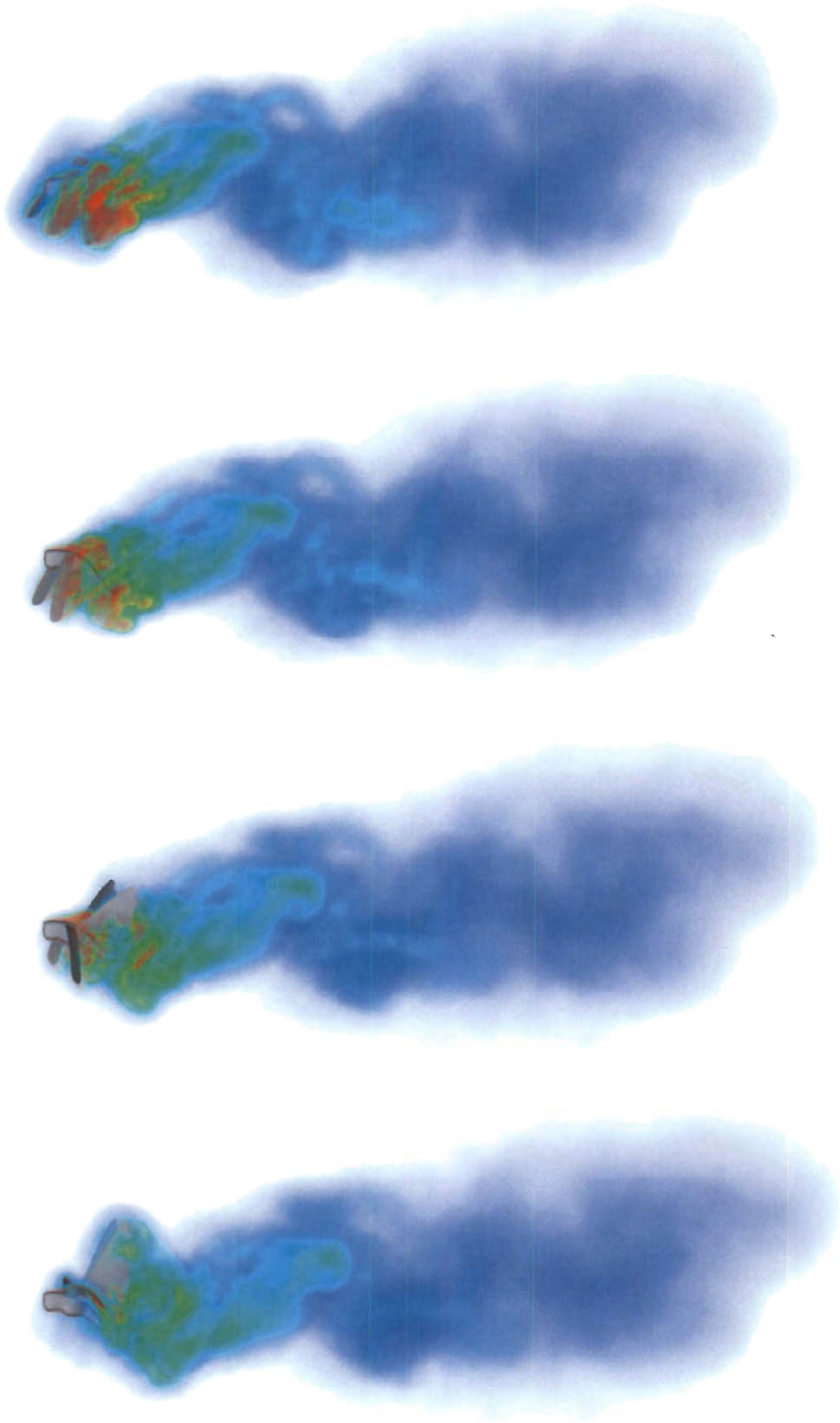
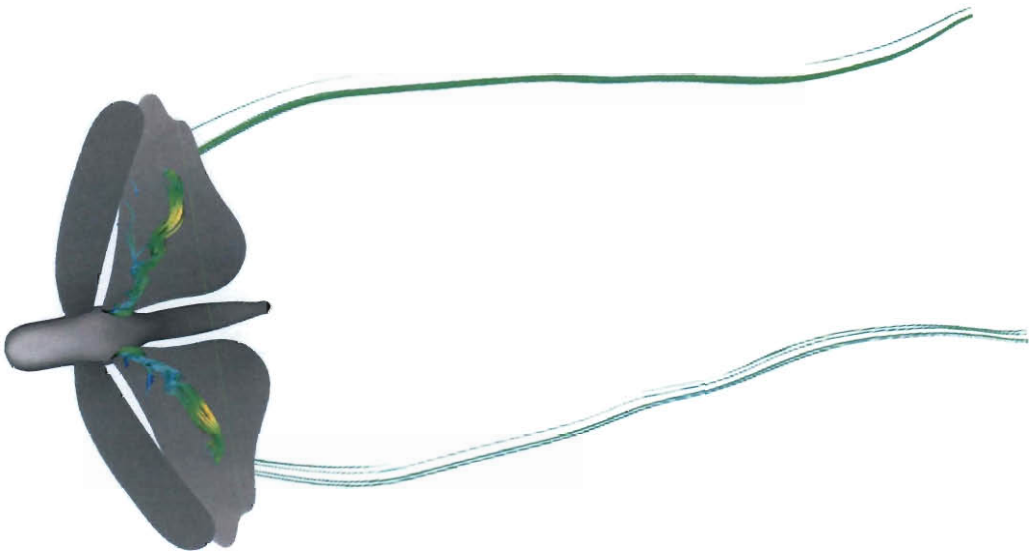
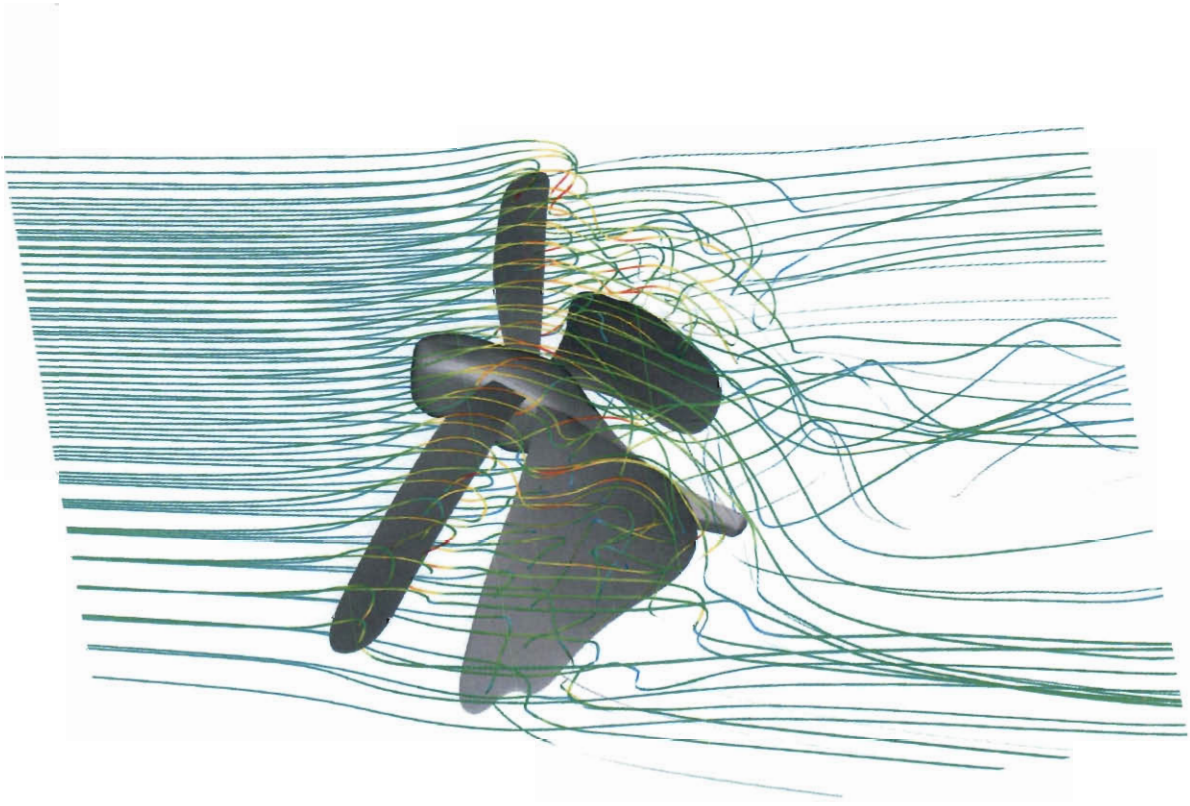
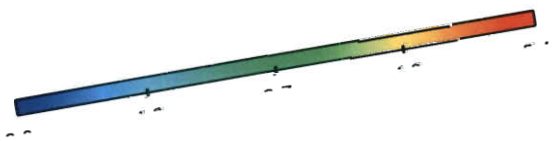


Figure 5.13: Volume rendering of vorticity for the last four approximately equally-spaced points during the second flapping cycle (top to bottom). Red indicates higher vorticity.







Chapter 6

Test Computation with Temporal and Spatial NURBS Representation

As discussed in [49], greater accuracy can be obtained through the use of higher-order basis functions, the advantages of which cannot be fully realized without their application to both the spatial and temporal representations. While the computations presented in Chapters 4 and 5 utilize higher-order basis functions for temporal interpolation, they use linear tetrahedral elements for the spatial representation. In contrast, the computation in this chapter uses higher-order basis functions for both spatial and temporal representation.

We preform a test computation that is representative of a flapping wing. To create a representative flapping wing, we use multiple 3D patches for spatial representation of the volume mesh, which is considerably different from unstructured volume meshes often used in finite element computations. While this test case is much simpler than those presented in previous chapters, it provides a model for testing of various parameters and forms a framework for future developments.

The setup for this test computation is presented in Section 6.1, and initial results

are then presented in Section 6.2.

6.1 Setup

While similar to previous chapters in that this computation also models a flapping wing, the setup, specifically the mesh, is quite different from that presented in previous chapters. In this section, the setup for this test computation is discussed beginning with the mesh generation in in Section 6.1.1. The wing motion for this computation is also discussed briefly in Section 6.1.2.

6.1.1 Mesh Generation

In this computation, we base our simple flapping–wing model on the FW of the locust. For simplicity, the curvature of the wing near the tip is eliminated resulting in a rectangular surface. We use quadratic B–splines to generate this surface with 3 control points in both spanwise and chordwise directions. Hexahedral patches are generated to form the volume mesh around the simple wing. In generating these patches, we once again seek to use the minimum number of control points necessary; for a quadratic B–spline volumetric patch without degeneration, this results in 27 control points in each patch. The simple (left) wing surface and 12 spatial patches of this mesh can be seen in Figure 6.1. We increase the refinement of the mesh by spatial knot insertion. After inserting knots, we arrive at the spatial-control mesh shown in Figure 6.2. This mesh consists of 1,536 control points, which includes control points that are co-located among patches, and is used as an input to both the mesh motion and fluid computations. For mesh motion and fluid computations, only one control value is used for the co-located control points. After knot insertion, the simple wing surface is defined by 6 and 4 control points in the spanwise and chordwise directions, respectively.

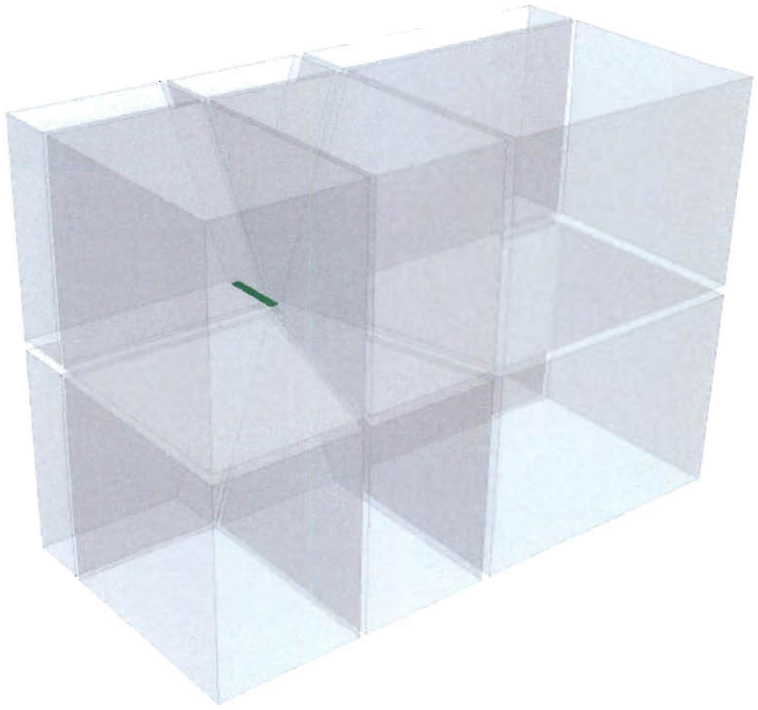


Figure 6.1: Undeformed mesh showing (left) wing surface and 12 spatial patches.

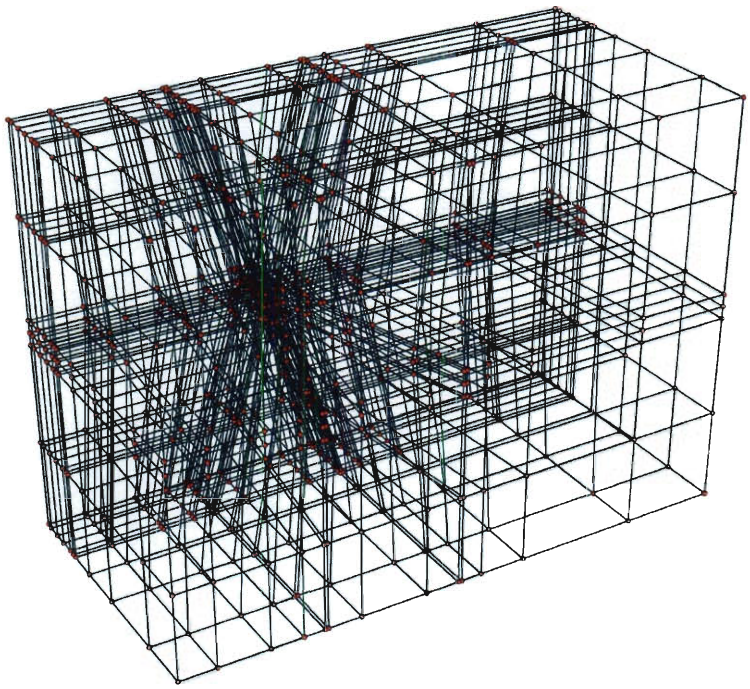


Figure 6.2: Undeformed control mesh after knot insertion used for mesh motion and fluid computations.

6.1.2 Prescribed Periodic Motion

The wing motion for this computation is based on the motion of the FW SS from Chapter 5. We elevate the order of the SS to 2nd order in both parametric directions to match the simple wing surface, which we previously generated. We insert knots as done in Section 6.1.1 to make the knots and number of control points of the surface we prescribe consistent with that of the simple wing surface. At each temporal-control point, the mesh is rotated to an angle of 27.4 degrees to account for the angle of the wing with respect to the freestream velocity.

After mesh motion, we use a similar process to that described in Section 5.1.3 to extract a single, periodic cycle. We perform this process on all control points in the mesh. Thus, continuity for all control points is maintained between cycles.

6.2 Results

This section presents results from this computation. In Section 6.2.1, the input parameters and results are shown. The fluid computation is presented in Section 6.2.2.

6.2.1 Mesh Motion

We compute the motion of the spatial-control points of the domain for all temporal-control points. As done in Chapters 4 and 5, we use subiterations to divide the steps between temporal-control into 20 smaller, linear steps. We then use 100 GMRES iterations for each subiteration to compute mesh motion.

We note that the number of control points used in knot insertion is not strictly required to be the same number of control points used for the fluid computation. Thus, less control points could be used to compute mesh motion and then knots could be inserted to increase spatial refinement for the fluid computation. We observe that if we over–reduce the degrees of freedom in the mesh motion, we increase the

potential for invalid solution–mesh motion. We overcome this problem by ensuring that a sufficient number of control points are used for the mesh motion computation.

6.2.2 Fluid Computation

After extracting a periodic cycle from the mesh computed in Section 6.2.1 with the process described in Section 6.1.2, we begin the fluid computation. For this fluid computation, we begin prescribed motion on the first time step. Meanwhile, we use a linear form to smoothly increase the inflow velocity from 0 to 2.4 m/s over the first 50 time steps. All parameters for this computation are the same as those described for Chapter 5 except that we use 30, 60, 30 and 60 GMRES iterations for each nonlinear iteration, respectively. Once we reach the full freestream velocity, we continue computing for the remainder of the cycle (225 steps more) to develop the flowfield to serve as a starting condition. As in the previous computation, the periodicity of the prescribed mesh motion allows us to use the output of one cycle as the input for the next.

After computing this starting condition, we increase the GMRES iterations to 30, 60, 90 and 120 for each nonlinear iteration and continue to compute. The results from these computations are shown in Figures 6.3 and 6.4.

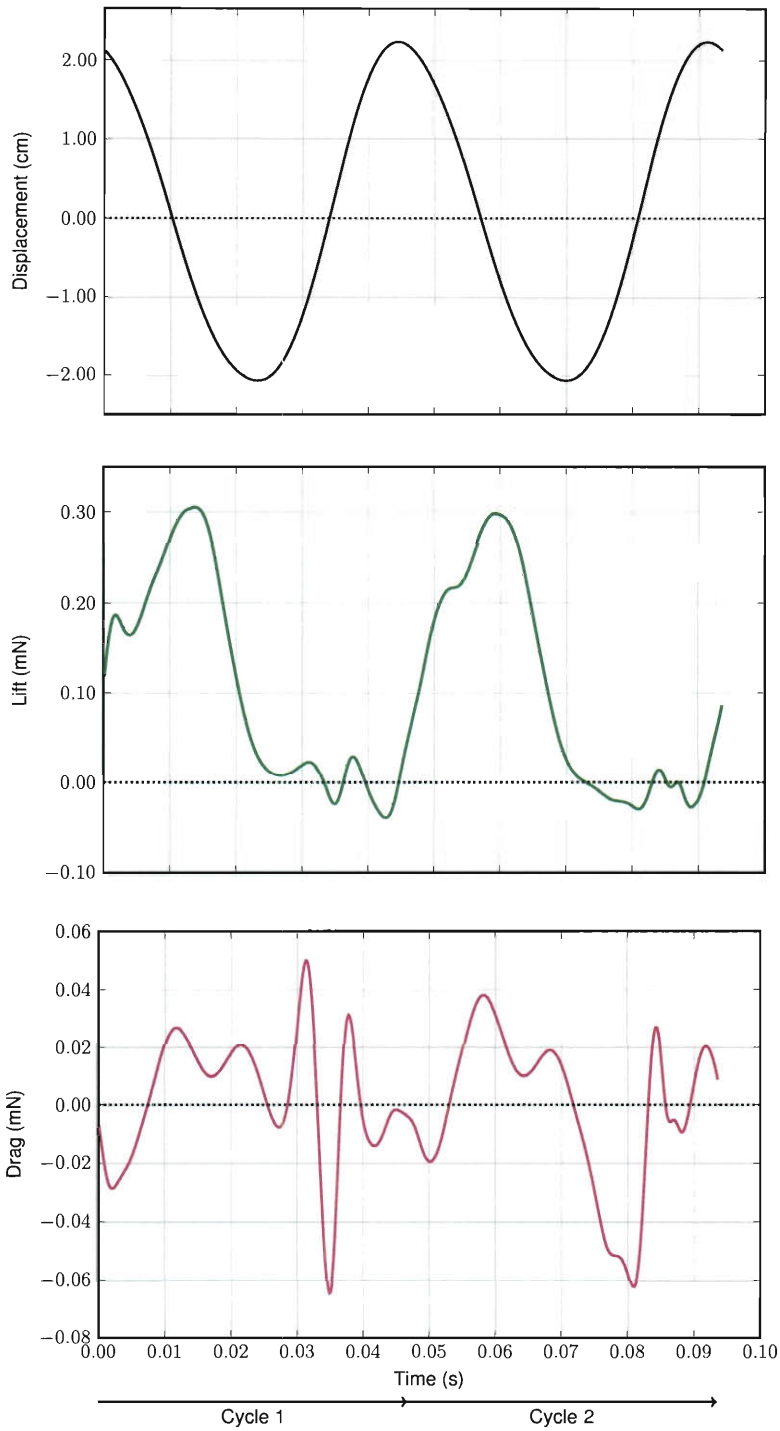


Figure 6.3: Wingtip displacement from the root chord (top), total lift force generated over two cycles (middle), total drag force, where a negative value indicates that thrust exceeds drag at that time (bottom). Note that the scales are different in the last two plots.

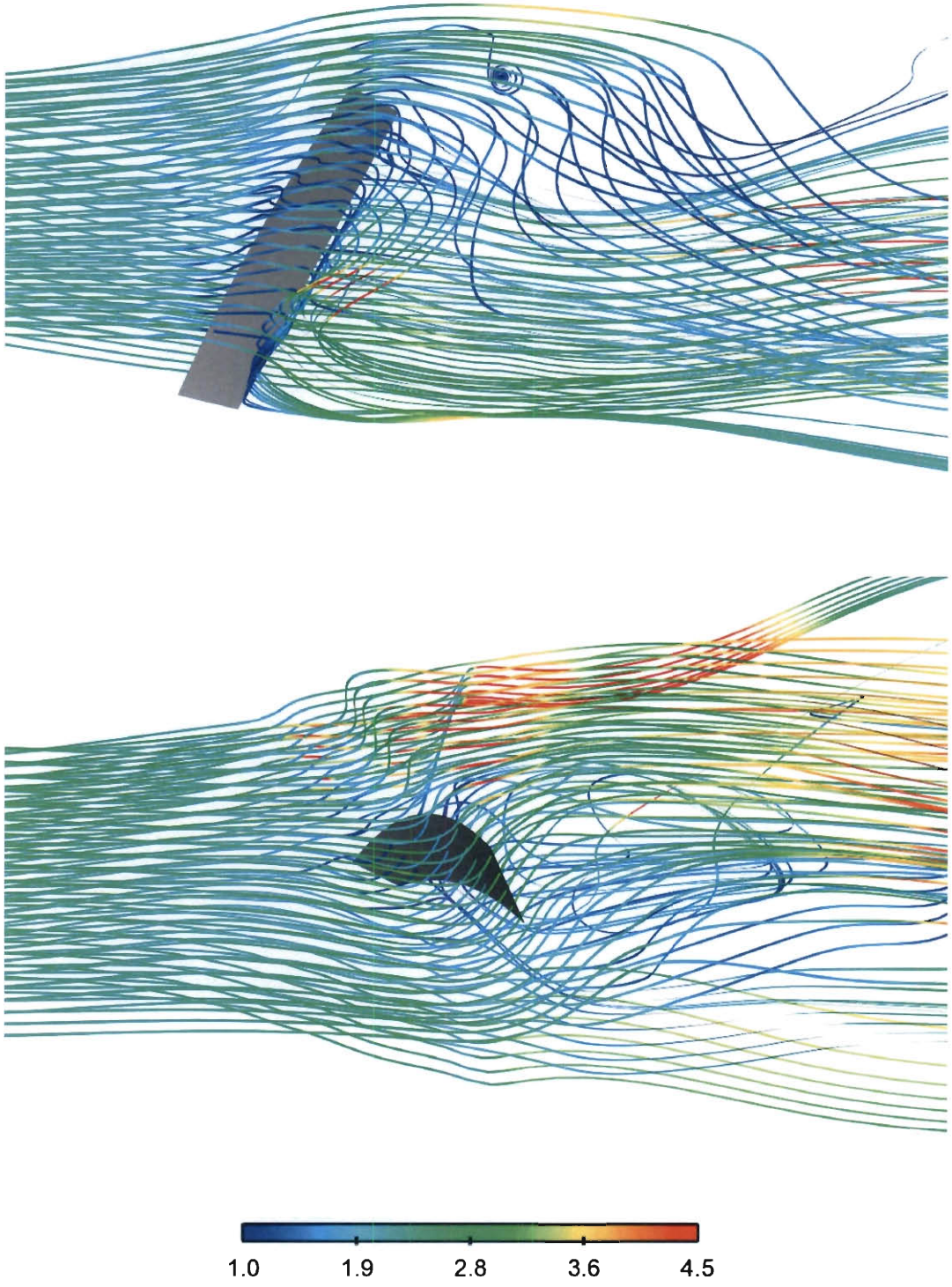


Figure 6.4: Streamlines colored by velocity in m/s at approximately 50% (top) and 75% (bottom) of the second flapping cycle period.

Chapter 7

Conclusions

This thesis has presented the special space–time computational techniques introduced recently by the T★AFSM for computation of flow problems with moving and deforming solid surfaces. The techniques have been designed in the context of the DSD/SST formulation, which is a general-purpose moving-mesh technique developed earlier by the T★AFSM for computation of flow problems with moving boundaries and interfaces. The main thrust of the special space–time techniques is using, in the space–time flow computations, NURBS basis functions for the temporal representation of the motion and deformation of the solid surfaces and also for the motion and deformation of the volume meshes computed. This provides a better temporal representation of the solid surfaces. It is also a more effective way of handling the volume-mesh motion as the solid surfaces deform.

These techniques have been applied in this thesis to computation of the aerodynamics of flapping wings, specifically locust wings, with prescribed motion and deformation. The time-dependent prescribed geometry of the wings comes from digital data extracted from the videos of the locust in a wind tunnel. Results from the preliminary computations and improved versions of those computations with even better temporal representation have been presented. Additionally, a flapping wing test computation with NURBS representation in both time and space has been pre-

sented. Using higher-order basis functions increases the accuracy, scope and efficiency of space–time computations, and how that can be done has been successfully demonstrated in this thesis, with emphasis on temporal representation.

Bibliography

[1] Science Daily, Secrets of insect flight revealed: Modeling the aerodynamic secrets of one of nature's most efficient flyers, 18 Sep 2009. `http://www.sciencedaily.com/releases/2009/09/090917144125.htm`.

[2] J. E. Akin, T. Tezduyar, M. Ungor, and S. Mittal. Stabilization parameters and Smagorinsky turbulence model. *Journal of Applied Mechanics*, 70:2–9, 2003.

[3] J. E. Akin and T. E. Tezduyar. Calculation of the advective limit of the SUPG stabilization parameter for linear and higher-order elements. *Computer Methods in Applied Mechanics and Engineering*, 193:1909–1922, 2004.

[4] D. E. Alexander. *Nature's Flyers: Birds, Insects, and the Biomechanics of Flight.* Johns Hopkins Univ Press, 2002.

[5] S. K. Aliabadi and T. E. Tezduyar. Parallel fluid dynamics computations in aerospace applications. *International Journal for Numerical Methods in Fluids*, 21:783–805, 1995.

[6] Y. Bazilevs and I. Akkerman. Large eddy simulation of turbulent Taylor–Couette flow using isogeometric analysis and the residual–based variational multiscale method. *Journal of Computational Physics*, 229:3402–3414, 2010.

[7] Y. Bazilevs, V. M. Calo, J. A. Cottrel, T. J. R. Hughes, A. Reali, and G. Scovazzi. Variational multiscale residual-based turbulence modeling for large eddy

simulation of incompressible flows. *Computer Methods in Applied Mechanics and Engineering*, 197:173–201, 2007.

[8] Y. Bazilevs, V. M. Calo, T. J. R. Hughes, and Y. Zhang. Isogeometric fluid–structure interaction: theory, algorithms, and computations. *Computational Mechanics*, 43:3–37, 2008.

[9] Y. Bazilevs, V. M. Calo, Y. Zhang, and T. J. R. Hughes. Isogeometric fluid–structure interaction analysis with applications to arterial blood flow. *Computational Mechanics*, 38:310–322, 2006.

[10] Y. Bazilevs, M.-C. Hsu, I. Akkerman, S. Wright, K. Takizawa, B. Henicke, T. Spielman, and T. E. Tezduyar. 3D simulation of wind turbine rotors at full scale. Part I: Geometry modeling and aerodynamics. *International Journal for Numerical Methods in Fluids*, 65:207–235, 2011.

[11] Y. Bazilevs and T. J. R. Hughes. NURBS-based isogeometric analysis for the computation of flows about rotating components. *Computational Mechanics*, 43:143–150, 2008.

[12] M. Behr, A. Johnson, J. Kennedy, S. Mittal, and T. Tezduyar. Computation of incompressible flows with implicit finite element implementations on the Connection Machine. *Computer Methods in Applied Mechanics and Engineering*, 108:99–118, 1993.

[13] M. Behr and T. Tezduyar. The Shear-Slip Mesh Update Method. *Computer Methods in Applied Mechanics and Engineering*, 174:261–274, 1999.

[14] A. N. Brooks and T. J. R. Hughes. Streamline upwind/Petrov-Galerkin formulations for convection dominated flows with particular emphasis on the incompressible Navier-Stokes equations. 32:199–259, 1982.

[15] L. Catabriga, A. L. G. A. Coutinho, and T. E. Tezduyar. Compressible flow SUPG parameters computed from element matrices. *Communications in Numerical Methods in Engineering*, 21:465–476, 2005.

[16] L. Catabriga, A. L. G. A. Coutinho, and T. E. Tezduyar. Compressible flow SUPG parameters computed from degree-of-freedom submatrices. *Computational Mechanics*, 38:334–343, 2006.

[17] A. Corsini, F. Rispoli, and T. E. Tezduyar. Stabilized finite element computation of NOx emission in aero-engine combustors. *International Journal for Numerical Methods in Fluids*, 65:254–270, 2011.

[18] J. A. Cottrell, T. Hughes, and Y. Bazilevs. *Isogeometric Analysis: Toward Integration of CAD and FEA*. John Wiley and Sons, Ltd., 2009.

[19] J. R. Helfer. *How to Know the Grasshoppers, Crickets, Cockroaches and Their Allies*. W.C. Brown, 1972.

[20] R. C. Herbert, P. G. Young, C. W. Smith, R. Wooton, and K. Evans. The hind wing of the desert locust (schisocerca gregaria forskål). III. A finite element analysis of a deployable structure. *Journal of Experimental Biology*, 203:2945–2955, 2000.

[21] M.-C. Hsu, Y. Bazilevs, V. M. Calo, T. E. Tezduyar, and T. J. R. Hughes. Improving stability of stabilized and multiscale formulations in flow simulations at small time steps. *Computer Methods in Applied Mechanics and Engineering*, 199:828–840, 2010.

[22] T. J. R. Hughes. Multiscale phenomena: Green's functions, the Dirichlet-to-Neumann formulation, subgrid scale models, bubbles, and the origins of stabilized methods. *Computer Methods in Applied Mechanics and Engineering*, 127:387–401, 1995.

[23] T. J. R. Hughes, J. A. Cottrell, and Y. Bazilevs. Isogeometric analysis: CAD, finite elements, NURBS, exact geometry, and mesh refinement. *Computer Methods in Applied Mechanics and Engineering*, 194:4135–4195, 2005.

[24] T. J. R. Hughes and A. A. Oberai. Calculation of shear stress in Fourier–Galerkin formulations of turbulent channel flows: projection, the Dirichlet filter and conservation. *Journal of Computational Physics*, 188:281–295, 2003.

[25] T. J. R. Hughes, A. A. Oberai, and L. Mazzei. Large eddy simulation of turbulent channel flows by the variational multiscale method. *Physics of Fluids*, 13:1784–1799, 2001.

[26] T. J. R. Hughes and G. Sangalli. Variational multiscale analysis: the fine-scale Green's function, projection, optimization, localization, and stabilized methods. *SIAM Journal of Numerical Analysis*, 45:539–557, 2007.

[27] M. Jensen. Biology and physics of locust flight. III. The aerodynamics of locust flight. *Philosophical Transactions of the Royal Society of London. Series B, Biological Sciences*, 239:511–552, 1956.

[28] A. A. Johnson and T. E. Tezduyar. Advanced mesh generation and update methods for 3D flow simulations. *Computational Mechanics*, 23:130–143, 1999.

[29] V. Kalro and T. E. Tezduyar. A parallel 3D computational method for fluid–structure interactions in parachute systems. *Computer Methods in Applied Mechanics and Engineering*, 190:321–332, 2000.

[30] G. Karypis and V. Kumar. A fast and high quality multilevel scheme for partitioning irregular graphs. *SIAM Journal of Scientific Computing*, 20:359–392, 1998.

[31] M. Lecoq. Smithsonian National Museum of Natural History: Forces of Change, Locust swarm. `http://forces.si.edu/elnino/exhibition_3f2.html`.

[32] J. H. McMasters. The flight of the bumblebee and related myths of entomological engineering. *American Scientist*, 77:164–169, 1989.

[33] S. Mittal and T. E. Tezduyar. Parallel finite element simulation of 3D incompressible flows – Fluid-structure interactions. *International Journal for Numerical Methods in Fluids*, 21:933–953, 1995.

[34] T. J. Mueller and J. D. DeLaurier. An overview of micro air vehicle aerodynamics. In T. J. Mueller, editor, *Fixed and Flapping Wing Aerodynamics for Micro Air Vehicle Applications*. AIAA, 2001.

[35] N. Phillips. ABC Science, Locust wings to inspire flying robots, 18 Sep 2009. `http://www.abc.net.au/news/stories/2009/09/18/2690317.htm`.

[36] D. J. Pines and F. Bohorquez. Challenges facing future micro-air-vehicle development. *Journal of Aircraft*, 43:290–305, 2006.

[37] J. R. H. Arnett. *American Insects: A Handbook of the Insects of America North of Mexico–2nd Ed.* CRC Press, 2000.

[38] F. Rispoli, A. Corsini, and T. E. Tezduyar. Finite element computation of turbulent flows with the discontinuity-capturing directional dissipation (DCDD). *Computers & Fluids*, 36:121–126, 2007.

[39] S. Sathe and T. E. Tezduyar. Modeling of fluid–structure interactions with the space–time finite elements: Contact problems. *Computational Mechanics*, 43:51–60, 2008.

[40] F. Shakib, T. J. R. Hughes, and Z. Johan. A new finite element formulation for computational fluid dynamics: X. The compressible euler and navier-stokes equations. *Comput. Methods Appl. Mech. and Engrg.*, 89:141–219, 1991.

[41] K. Stein, R. Benney, T. Tezduyar, and J. Potvin. Fluid–structure interactions of a cross parachute: Numerical simulation. *Computer Methods in Applied Mechanics and Engineering*, 191:673–687, 2001.

[42] K. Takizawa, J. Christopher, T. E. Tezduyar, and S. Sathe. Space–time finite element computation of arterial fluid–structure interactions with patient-specific data. *International Journal for Numerical Methods in Biomedical Engineering*, 26:101–116, 2010.

[43] K. Takizawa, B. Henicke, A. Puntel, T. Spielman, and T. E. Tezduyar. Space–time computational techniques for the aerodynamics of flapping wings. *Journal of Applied Mechanics*, to appear, 2011.

[44] K. Takizawa, B. Henicke, T. E. Tezduyar, M.-C. Hsu, and Y. Bazilevs. Stabilized space–time computation of wind-turbine rotor aerodynamics. *Computational Mechanics*, published online, DOI: 10.1007/s00466-011-0589-2, March 2011.

[45] K. Takizawa, C. Moorman, S. Wright, J. Christopher, and T. E. Tezduyar. Wall shear stress calculations in space–time finite element computation of arterial fluid–structure interactions. *Computational Mechanics*, 46:31–41, 2010.

[46] K. Takizawa, C. Moorman, S. Wright, J. Purdue, T. McPhail, P. R. Chen, J. Warren, and T. E. Tezduyar. Patient-specific arterial fluid–structure interaction modeling of cerebral aneurysms. *International Journal for Numerical Methods in Fluids*, 65:308–323, 2011.

[47] K. Takizawa, C. Moorman, S. Wright, T. Spielman, and T. E. Tezduyar. Fluid–structure interaction modeling and performance analysis of the Orion spacecraft parachutes. *International Journal for Numerical Methods in Fluids*, 65:271–285, 2011.

[48] K. Takizawa, T. Spielman, and T. E. Tezduyar. Space–time FSI modeling and dynamical analysis of spacecraft parachutes and parachute clusters. *Computational Mechanics*, published online, DOI: 10.1007/s00466-011-0590-9, April 2011.

[49] K. Takizawa and T. E. Tezduyar. Multiscale space–time fluid–structure interaction techniques. *Computational Mechanics*, published online, DOI: 10.1007/s00466-011-0571-z, February 2011.

[50] K. Takizawa, S. Wright, C. Moorman, and T. E. Tezduyar. Fluid–structure interaction modeling of parachute clusters. *International Journal for Numerical Methods in Fluids*, 65:286–307, 2011.

[51] T. Tezduyar, S. Aliabadi, M. Behr, A. Johnson, and S. Mittal. Parallel finite-element computation of 3D flows. *Computer*, 26(10):27–36, 1993.

[52] T. Tezduyar and Y. Osawa. Fluid–structure interactions of a parachute crossing the far wake of an aircraft. *Computer Methods in Applied Mechanics and Engineering*, 191:717–726, 2001.

[53] T. E. Tezduyar. Stabilized finite element formulations for incompressible flow computations. *Advances in Applied Mechanics*, 28:1–44, 1992.

[54] T. E. Tezduyar. Computation of moving boundaries and interfaces and stabilization parameters. *International Journal for Numerical Methods in Fluids*, 43:555–575, 2003.

[55] T. E. Tezduyar. Finite element methods for fluid dynamics with moving boundaries and interfaces. In E. Stein, R. D. Borst, and T. J. R. Hughes, editors, *Encyclopedia of Computational Mechanics*, Volume 3: Fluids, chapter 17. John Wiley & Sons, 2004.

[56] T. E. Tezduyar. Finite elements in fluids: Stabilized formulations and moving boundaries and interfaces. *Computers & Fluids*, 36:191–206, 2007.

[57] T. E. Tezduyar, M. Behr, and J. Liou. A new strategy for finite element computations involving moving boundaries and interfaces – the deforming-spatial-domain/space–time procedure: I. The concept and the preliminary numerical tests. *Computer Methods in Applied Mechanics and Engineering*, 94(3):339–351, 1992.

[58] T. E. Tezduyar, M. Behr, S. Mittal, and J. Liou. A new strategy for finite element computations involving moving boundaries and interfaces – the deforming-spatial-domain/space–time procedure: II. Computation of free-surface flows, two-liquid flows, and flows with drifting cylinders. *Computer Methods in Applied Mechanics and Engineering*, 94(3):353–371, 1992.

[59] T. E. Tezduyar, S. Mittal, S. E. Ray, and R. Shih. Incompressible flow computations with stabilized bilinear and linear equal-order-interpolation velocity-pressure elements. *Computer Methods in Applied Mechanics and Engineering*, 95:221–242, 1992.

[60] T. E. Tezduyar and Y. Osawa. Finite element stabilization parameters computed from element matrices and vectors. *Computer Methods in Applied Mechanics and Engineering*, 190:411–430, 2000.

[61] T. E. Tezduyar and Y. J. Park. Discontinuity capturing finite element formulations for nonlinear convection-diffusion-reaction equations. *Computer Methods in Applied Mechanics and Engineering*, 59:307–325, 1986.

[62] T. E. Tezduyar and S. Sathe. Modeling of fluid–structure interactions with the space–time finite elements: Solution techniques. *International Journal for Numerical Methods in Fluids*, 54:855–900, 2007.

[63] T. E. Tezduyar, S. Sathe, T. Cragin, B. Nanna, B. S. Conklin, J. Pausewang, and M. Schwaab. Modeling of fluid–structure interactions with the space–time

finite elements: Arterial fluid mechanics. *International Journal for Numerical Methods in Fluids*, 54:901–922, 2007.

[64] T. E. Tezduyar, S. Sathe, J. Pausewang, M. Schwaab, J. Christopher, and J. Crabtree. Fluid–structure interaction modeling of ringsail parachutes. *Computational Mechanics*, 43:133–142, 2008.

[65] T. E. Tezduyar, S. Sathe, J. Pausewang, M. Schwaab, J. Christopher, and J. Crabtree. Interface projection techniques for fluid–structure interaction modeling with moving-mesh methods. *Computational Mechanics*, 43:39–49, 2008.

[66] T. E. Tezduyar, S. Sathe, M. Schwaab, and B. S. Conklin. Arterial fluid mechanics modeling with the stabilized space–time fluid–structure interaction technique. *International Journal for Numerical Methods in Fluids*, 57:601–629, 2008.

[67] T. E. Tezduyar, S. Sathe, and K. Stein. Solution techniques for the fully-discretized equations in computation of fluid–structure interactions with the space–time formulations. *Computer Methods in Applied Mechanics and Engineering*, 195:5743–5753, 2006.

[68] T. E. Tezduyar, M. Schwaab, and S. Sathe. Sequentially-Coupled Arterial Fluid–Structure Interaction (SCAFSI) technique. *Computer Methods in Applied Mechanics and Engineering*, 198:3524–3533, 2009.

[69] T. E. Tezduyar, K. Takizawa, T. Brummer, and P. R. Chen. Space–time fluid–structure interaction modeling of patient-specific cerebral aneurysms. *International Journal for Numerical Methods in Biomedical Engineering*, published online, DOI: 10.1002/cnm.1433, February 2011.

[70] T. E. Tezduyar, K. Takizawa, C. Moorman, S. Wright, and J. Christopher. Multiscale sequentially-coupled arterial FSI technique. *Computational Mechanics*, 46:17–29, 2010.

[71] T. E. Tezduyar, K. Takizawa, C. Moorman, S. Wright, and J. Christopher. Space–time finite element computation of complex fluid–structure interactions. *International Journal for Numerical Methods in Fluids*, 64:1201–1218, 2010.

[72] R. Torii, M. Oshima, T. Kobayashi, K. Takagi, and T. E. Tezduyar. Computer modeling of cardiovascular fluid–structure interactions with the Deforming-Spatial-Domain/Stabilized Space–Time formulation. *Computer Methods in Applied Mechanics and Engineering*, 195:1885–1895, 2006.

[73] R. Torii, M. Oshima, T. Kobayashi, K. Takagi, and T. E. Tezduyar. Fluid–structure interaction modeling of aneurysmal conditions with high and normal blood pressures. *Computational Mechanics*, 38:482–490, 2006.

[74] R. Torii, M. Oshima, T. Kobayashi, K. Takagi, and T. E. Tezduyar. Numerical investigation of the effect of hypertensive blood pressure on cerebral aneurysm — Dependence of the effect on the aneurysm shape. *International Journal for Numerical Methods in Fluids*, 54:995–1009, 2007.

[75] R. Torii, M. Oshima, T. Kobayashi, K. Takagi, and T. E. Tezduyar. Fluid–structure interaction modeling of a patient-specific cerebral aneurysm: Influence of structural modeling. *Computational Mechanics*, 43:151–159, 2008.

[76] R. Torii, M. Oshima, T. Kobayashi, K. Takagi, and T. E. Tezduyar. Fluid–structure interaction modeling of blood flow and cerebral aneurysm: Significance of artery and aneurysm shapes. *Computer Methods in Applied Mechanics and Engineering*, 198:3613–3621, 2009.

[77] R. Torii, M. Oshima, T. Kobayashi, K. Takagi, and T. E. Tezduyar. Influence of wall thickness on fluid–structure interaction computations of cerebral aneurysms. *International Journal for Numerical Methods in Biomedical Engineering*, 26:336–347, 2010.

[78] R. Torii, M. Oshima, T. Kobayashi, K. Takagi, and T. E. Tezduyar. Role of 0D peripheral vasculature model in fluid–structure interaction modeling of aneurysms. *Computational Mechanics*, 46:43–52, 2010.

[79] S. Walker, A. Thomas, and G. Taylor. Deformable wing kinematics in the desert locust: how and why do camber, twist and topography vary through the stroke? *Journal of the Royal Society Interface*, 6:735–747, 2009.

[80] T. Weis-Fogh. Biology and physics of locust flight. II. Flight performance of the desert locust (schistocerca gregaria). *Philosophical Transactions of the Royal Society of London. Series B, Biological Sciences*, 239:459–510, 1956.

[81] T. Weis-Fogh. Biology and physics of locust flight. IV. Notes on sensory mechanisms in locust flight. *Philosophical Transactions of the Royal Society of London. Series B, Biological Sciences*, 239:553–584, 1956.

[82] T. Weis-Fogh and M. Jensen. Biology and physics of locust flight. I. Basic principles in insect flight. A critical review. *Philosophical Transactions of the Royal Society of London. Series B, Biological Sciences*, 239:415–458, 1956.

[83] R. J. Wootton, K. E. Evans, R. Herbert, and C. W. Smith. The hind wing of the desert locust (schisocerca gregaria forskål). I. functional morphology and mode of operation. *Journal of Experimental Biology*, 203:2921–2931, 2000.

[84] T. Y. Wu. Fish swimming and bird/insect flight. *Annual Review of Fluid Mechanics*, 43:25–58, 2011.

[85] J. Young, S. Walker, R. Bomphrey, G. Taylor, and A. Thomas. Details of insect wing design and deformation enhance aerodynamic function and flight efficiency. *Science*, 325:1549–1552, 2009.